Tampereen yliopisto

Vesa Kyllönen

TIETOLIIKENNELAITTEIDEN ESIOHJELMOINNIN AUTOMATISOINTI

Diplomityö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Prof. Mikko Valkama
Heinäkuu 2020

TIIVISTELMÄ

Vesa Kyllönen: Tietoliikennelaitteiden esiohjelmoinnin automatisointi
Diplomityö
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Heinäkuu 2020

Operaattoriverkko koostuu useisiin eri käyttötarkoituksiin tarkoitetuista verkkolaitteista. Käytännössä näiden verkkolaitteiden asetukset tarvitsevat etukäteen tehtäviä määrittelyitä ennen kuin ne pystytään turvallisesti ja hallitusti tuomaan osaksi suurempaa tietoverkkoa. Teknisesti monimutkaisen verkon asennuksien osaaminen vaatii myös asentajilta laajaa osaamista verkkolaitteiden asennuksen ohessa tehtävään konfigurointiin, jonka myötä riittävän ohjeistuksen ja koulutuksen ylläpito maksaa paljon työnantajalle.

Tässä työssä suunnitellaan ja rakennetaan laite, jonka tehtävänä on automaattisesti esiohjelmoida verkkolaitteita niiden RS232 -konsoliportin kautta. Asentajan päivittäisen työkalun tehtävänä on määritellä verkkolaitteelle operaattorin antamat verkkoasetukset, jotta laite voidaan turvallisesti lisätä osaksi suurempaa verkkoa. Luodun prototyypin tehtävänä on osata tehdä Huawei VRP -pohjaisen verkkolaitteen konfiguraation määrittelyt, testaus sekä nollaus.

Prototyypin alustana käytetään Raspberry Pi -tietokonetta, jonka laaja dokumentaatio ja ohjelmistotuki tekivät siitä helpon valinnan. Ohjelmiston pohjana käytettiin Raspbian -käyttöjärjestelmää ja Python -ohjelmointikieltä. PySerial -sarjaliikenneportin ohjaukseen määritellyn moduulin avulla saatiin valmiit käskyt verkkolaitteen kanssa kommunikointiin. Toteutetun ohjelman kommunikointi verkkolaitteen kanssa oli aluksi hieman hankalaa verkkolaitteen konsoliportista lähettämien tulosteiden tulkitsemiseksi, mutta muutaman käyttöjärjestelmään tehtävän määrittelyn ansiosta saatiin ohjelmiston logiikka toimimaan oikein jokaisessa tilanteessa.

Tuloksena oli toimiva prototyyppi, jonka pohjalta on helppo lähteä laajentamaan laitetukea eri valmistajien käyttöjärjestelmille.

Avainsanat: Operaattoriverkko, Metro Ethernet, sarjamuotoinen tiedonsiirto, Python, RS232

ABSTRACT

Vesa Kyllönen: Automation of computer networking device preconfiguration
Master's thesis
Tampere University
Master's Degree Programme in Information technology
July 2020

An operator network consists of network devices intended for several different purposes. In practice, the settings of these network devices need to be defined in advance before they can be securely added as part of larger data network. Performing installations in technically complex network is demanding work and maintaining know-how of field engineers is continuous process. Creating and maintaining installation instructions in technically complex network is continuous task. Field engineers need to have knowledge how to configure different types of devices while new devices are constantly introduced.

The task is to design and create a device which can configure other network devices. This preconfigurator is meant to be fully automated and a daily tool for field engineer. It connects to other devices RS232 based console port and sends preconfiguration settings to the network device.

Created prototype is to be able to configure, test and reset the Huawei VRP-based network device. The Raspberry Pi computer was chosen platform for the prototype because of its extensive documentation and software support. Raspbian is operating system used on Raspberry Pi and Python language was used for writing the software. PySerial module was used to control the serial communication port and it provides ready-made commands to communicate with the network device.

The communication between the program and the network device was initially a bit tricky, because of logging messages received from console port, but programs logic started working after changing some system settings in Huawei VRP.

The result was a working prototype that makes it easy to add more hardware support for operating systems from different manufacturers.

Keywords: Carrier network, Metro Ethernet, serial communication, Python, RS232

ALKUSANAT

Aloittaessani opiskeluni Tampereen teknillisessä yliopistossa olin aika varma etten koskaan tule sieltä valmistumaan. Muutaman vuoden suunnan etsiminen sai minut kuitenkin tajuamaan että opinnot voivat perustua omiin harrastuksiin eikä siihen mitä aikaisemmin on opiskellut. Tajutessani tietoliikennetekniikan olevan se ala minkä parissa haluaa aloittaa uransa oli motivaatiota suorittaa pakolliset kurssit alta pois. Edessä vain oli enää se suuri pelottava diplomityö, jonka kirjoittamisesta olin jo saanut ensimakua AMK:n opinnäytetyön muodossa. Pienen vastoinkäymisen jälkeen pääsin kuitenkin töihin nykyiseen työpaikkaani Telialle, joka osoittautui hyväksi työpaikaksi ja juuri sellaisiin tehtäviin mihin toivoin päätyväni.

Haluankin osoittaa kiitokseni ohjaajalle Kalle Valkamalle työn aiheesta sekä tarkastajalle professori Mikko Valkamalle kärsivällisyydestä työn etenemisen suhteen.

Erityiskiitoksen haluan antaa Signaalinkäsittelyn ja tietoliikennetekniikan ammattiainekerholle TeLElle, Kelvoille ystävilleni sekä tietysti rakkaalle Teekerholle unohtumattomista kokemuksista teekkarielämässä ja sen ulkopuolella.

Pirkkalassa, 1. heinäkuuta 2020

Vesa Kyllönen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

AMOLED	Active-Matrix Organic Light-Emitting Diode
APB	Advanced Peripheral Bus
ASCII	American Standard Code for Information Interchange
DCE	A data circuit-terminating equipment
DE9	9 -pinninen elektroniikkaliitin
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
DTE	Data terminal equipment
FIFO	First In First Out
FTTx	Fiber To The X, esimerkiksi FTTB, joka tarkoittaa kuidun tuomista rakennukseen asti (Building)
GPIO	General-Purpose Input/Output
HDMI	High-Definition Multimedia Interface
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IOS	Internetwork Operating System
MAC	Media Access Control (Address)
MAN	Metropolitan Area Network
MPLS	Multiprotocol Label Switching
MTU	Maximum Transmission Unit
PISO	Parallel In Serial Out, rinnanmuotoista signaalia sisään, sarjaa ulos
RS232	Recommended Standard 232
RSA	Rivest-Shamir-Adleman kryptausjärjestelmä
SD	Secure Digital
SFP	Small form-factor, Verkkolaitteissa käytetty irroitettava liitinmoduuli
SFTP	Secure File Transfer Protocol
SoC	System on Chip

SSH	Secure SHell
T1	Transmission System 1, Euroopassa oma E1
TFTP	Trivial File Transfer Protocol
TIA	Telecommunications Industry Association
TTL	Transistor–transistor logic
UART	Universal asynchronous receiver-transmitter
USB	Universal Serial Bus
VRP	Versatile Routing Platform
WAN	Wide Area Network

1 JOHDANTO

Diplomityössäni suunnittelen ja toteutan kytkimiä ja reitittimiä ohjelmoivan laitteen, jota käytetään tietoliikennelaitteiden asennuksen yhteydessä. Tietoliikenneverkkoa rakennettaessa käytetään usein useamman valmistajan kytkimiä ja reitittimiä. Näitä ohjelmoiva laite, tästä eteenpäin käytetään termiä esiohjelmoija, toimii laitekantariippumattomana työkaluna asennuksen yhteydessä. Tarkoituksena onkin että esiohjelmoija toimii asennettavan laitteen konfiguroivana työkaluna, ettei asentajan tarvitse osata tämän käyttöliittymän kieltä eikä tarvitse yhdistää tietokonetta verkkolaitteen konsoliporttiin asetusten määrittämistä varten. Päämääränä on toteuttaa prototyyppi, joka toimii yhtenä asentajan työkaluna päivittäisessä käytössä. Prototyypin on tarkoitus olla demolaite, joka testaa yhden valmistajan verkkokäyttöjärjestelmän kanssa kommunikointia eikä sen tarvitse olla valmis tuote. Tämä työ on tehty Telia Finland Oyj:lle ja työskentelen runkoverkon ylläpitoon liittyvien asioiden parissa. Olen päivittäin yhteydessä verkkolaitteita asentavien teknikkojen kanssa eli tunnen aiheen yleisimmät ongelmat ja haasteet. Operaattoritason verkon rakentamisessa käytetään monipuolisesti eri tyyppisiä laitteita, joiden esiohjelmoimiseen tarvitaan monipuolisesti osaamista eri ohjelmistojen käskykannan mukaan. Esiasennusta hankaloittavat myös uusien ohjelmistopäivityksien mukana muuttuvat käskykannat. Konsolihallinnan muodostamisessa on yleisesti ongelmia eri valmistajien käyttämien standardien mukaan, jolloin asentajan täytyy tietää minkälaista konsolikaapelia käyttää esiasennuksessa. Varsinkin ohjeistuksen ylläpitäminen sekä asentajien suuri vaihtuvuus on luonut tarpeen asennusprosessin suoraviivaistamiseen mahdollisuuksien mukaan. Esiohjelmoijan on tarkoitus poistaa ohjelmoiminen asentajan rutiinityötehtävistä kokonaan, mikä tuo asennukseen sekä ajallista säästöä, sekä vähentää virheistä aiheutuvia kustannuksia. Alustana käytän Raspberry Pi -kehitysalustaa, jonka yleisyys ja laaja tukiverkosto mahdollistaa esiohjelmoijan toteuttamisen ja jatkokehityksen. Ohjelmistona käytetään alustalle muokattua Debian -pohjaista Raspbian linux distribuutiota ja itse ohjelma on toteutettu Python -ohjelmointikielellä. Prototyypissä on tarkoitus käyttää DE9 -portillista RS232 yhteensopivaa liitintä, jonka avulla voidaan luoda konsoliyhteys lähes mihin tahansa tietoliikennepohjaiseen laitteeseen ja muokata tämän asetuksia.

2 OPERAATTORIVERKKO

Tässä luvussa kerron miten nykyaikainen operaattoriverkko pystyy jakamaan palveluita loppukäyttäjälle. Aluksi kerron kuinka operaattoriverkko on 2000 -luvun aikana muuttunut Internetin kehittymisen myötä ja kuinka lähiverkkotekniikat ovat mahdollistaneet yritysten ja kuluttajien saumattoman verkostoitumisen. Lopuksi perehdyn tarkemmin kuinka valtakunnallinen operaattoriverkko fyysisesti rakentuu ja mitä uusien tietoliikennelaitteiden tuominen verkkoon käytännössä vaatii.

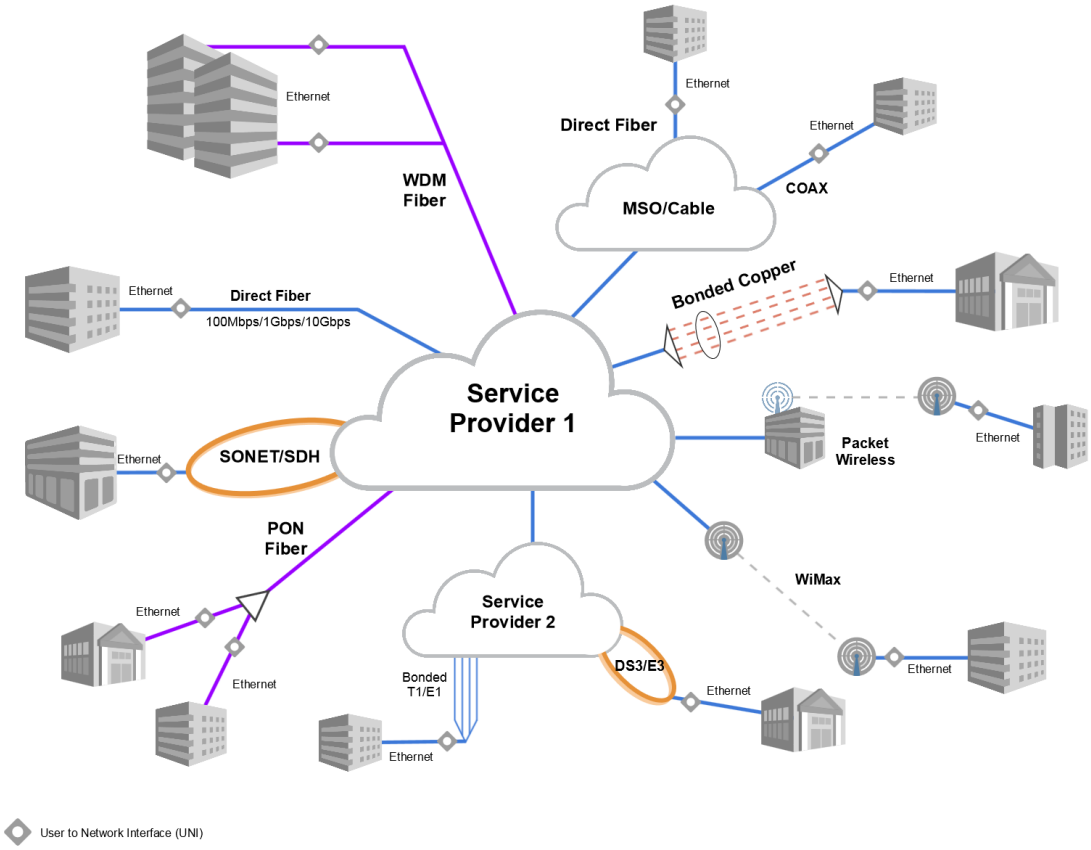
2.1 T-carrier ja Ethernet

Ethernet -tekniikka on alun perin tarkoitettu käytettäväksi paikallisen lähiverkon muodostamiseen. Siirryttäessä lähiverkosta toiseen on pitkien etäisyyksien takia jouduttu vaihtamaan erilaiseen tekniikkaan ja tällaisissa tapauksissa on haluttu hyödyntää olemassa olevia yhteyksiä, kuten esimerkiksi puhelinlinjoja. Puheluita siirtämään eri keskusten välillä käytettiin T-carrier -yhteyksiä, joiden tiedonsiirtokapasiteetti määräytyi käytettävien linjojen mukaan. Nämä linjat ovat aikasidonnaisia ja jokaista kanavaa pystyi varaamaan vain yksi tilaaja kerrallaan. Euroopassa käytössä ollut E1 yhteys pystyi siirtämään 24 samanaikaista puheyhteyttä samaan aikaan ja yhden äänikanavan teoreettinen tiedonsiirtonopeus on 64 kilobittiä, joka on myös puhelinverkkomodeemien suurin mahdollinen tiedonsiirtonopeus. Internetin yleistymisen myötä kaistantarve sekä asiakkaiden määrä nousi nopeasti, jolloin lisäkapasiteettia jouduttiin jatkuvasti lisäämään rakentamalla uusia T-carrier yhteyksiä. Operaattoreiden ratkaisu tähän pulmaan löytyi jo olemassa olevasta Ethernet -tekniikasta, jonka etuna esimerkiksi oli ettei siirtokaistaa tarvinnut varata yhdelle yhteydelle kerrallaan. Siirrettävä tieto pilkotaan ja paketoidaan MTU -paketeiksi, jotka sitten siirretään vastapäähän ilman reittisidonnaisuutta. Ethernetin etuna on että sen yli voidaan siirtää T-carrier yhteyksiä, jolloin asiakkaiden ei tarvitse luopua muuten toimivasta tietojärjestelmästään. [1]

2.2 Carrier Ethernet palvelut

Carrier Ethernetillä tarkoitetaan joukkoa Ethernet -palveluita, joissa sovelletaan Ethernet -tekniikan tuomia etuja MAN -verkossa. Asiakkaana voi toimia suuri organisaatio usealla toimipisteellä tai yksittäinen kuluttaja. Alunperin Carrier Ethernetillä pyrittiin ratkaisemaan

suurten yritysten ongelma yhdistää toimipisteensä lähiverkot toisiinsa, sekä ratkaisemaan vanhojen WAN -verkkojen kustannus ja skaalautuvuus ongelmia. Kuvassa 2.1 esitellään miten Ethernet mahdollistaa vanhempia tekniikoita hyödyntäen asiakasliitynnän palvelun-tarjoajalle.



Kuva 2.1. *Eri tiedonsiirtotekniikoiden hyödyntäminen ethernetpohjaisessa operaattoriverkossa. [2]*

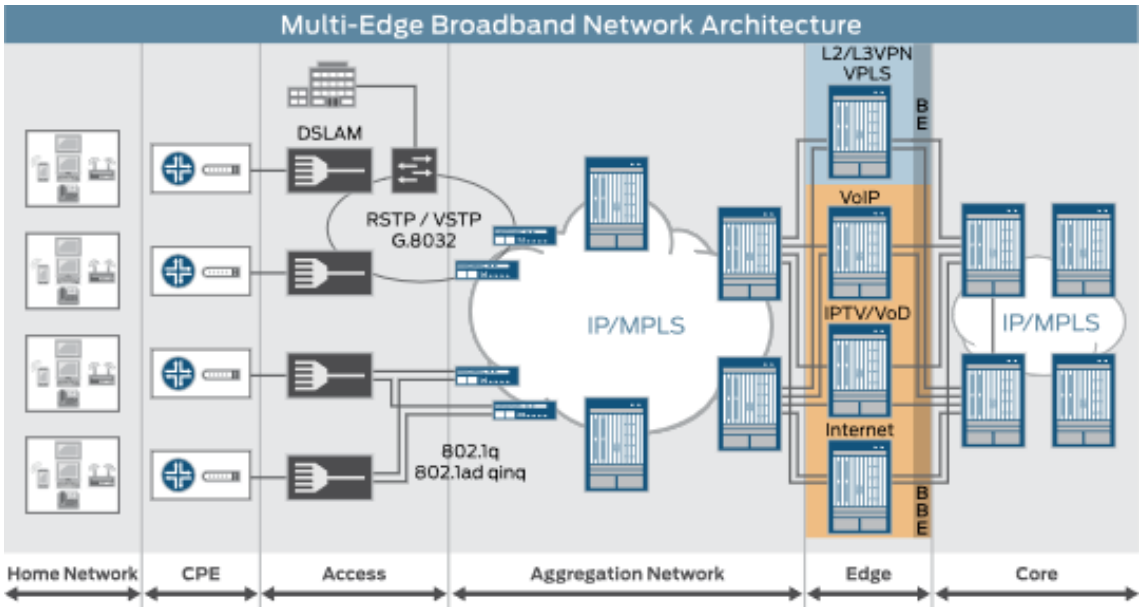
Metro Ethernet palveluiden kehittämisestä vastaa voittoa tavoittelematon taho Metro Ethernet Forum, joka koostuu operaattoreista, laite- ja ohjelmistovalmistajista ja muista tietoliikennealan yrityksistä. Osalliset yhdessä kehittävät määritelmät, joilla palvelut saadaan pysymään yhteensopivina keskenään. Tehtävänä on toimia yhdistävänä organisaationa ja tarjota jäsenilleen mahdollisuuksia vaikuttaa kehitykseen. Lisäksi foorumi tarjoaa verkkojen ja verkkolaitteiden sertifiointia ja henkilökunnan koulutusta. Metro Ethernet palvelut tuodaan operaattoriverkoissa tilaajan saataville omissa virtuaaliverkoissaan, jolloin tilaaja tai tilaajan toimipisteet ovat loogisesti eriytetyt muusta verkosta. Näin toteutetun verkon hyötyinä ovat helppokäyttöisyys, skaalautuvuus sekä matalat ylläpitokustannukset. Tilaajana voi toimia yksittäinen kuluttaja tai suuri yritysverkko useammalla toimipisteellä.

MEF:in standardointien ansiosta operaattorit voivat tarjota palveluitaan toisten operaattoreiden verkoissa, ilman että omistava taho suoraan näkee mitä yhteydellä siirretään. Yrityksien eri toimipisteiden sijainnit voivat kuitenkin estää saman operaattorin palveluiden

täyden hyödyntämisen, jonka takia heidän täytyy ostaa kahdelta operaattorilta palveluita yhdistämään kaksi toimipistettään. MEF on asettanut määritteet minkä mukaan operaattorit ja asiakkaat voivat toteuttaa lähiverkon useamman toimipisteen välillä. Kuluttajayhteyksien vuokraaminen operaattoreiden kesken on myös mahdollista, mutta normaalissa kuluttajakäytössä voi olla edullisinta hankkia yhteys suoraan operaattorilta. [3]

2.3 Metro Ethernet

Metro Ethernet -termillä tarkoitetaan yleisesti valtakunnallista operaattoriverkkoa, joka on toteutettu Ethernet -tekniikalla. Metro Ethernet -pohjainen operaattoriverkko koostuu useasta eri osa-alueesta; Coresta, Edgestä, Aggregation verkosta sekä Access -kuluttajaliitynnästä. Nämä verkon osat muodostavat puumaisen rakenteen ja mahdollistavat palveluiden siirron loppukäyttäjälle. Kuvassa 2.2 on esitelty Juniperin useamman palvelun arkkitehtuuria aina asiakkaalle asti.



Kuva 2.2. *Tyypillinen useamman palvelun operaattoriverkon ratkaisu [4]*

2.3.1 Core ja Edge

Core -verkko mahdollistaa palveluiden reitittämisen operaattoriverkossa. Itse palvelut tuodaan Edge -reitittimiltä operaattoriverkkoon asiakkaiden saataville. Varsinaisen aggregation verkon eli palveluiden siirtoverkon muodostavat Metro -reitittimet, jotka mahdollistavat palveluiden siirron asiakkaille. Kuvasta 2.2 voidaan havaita miten esimerkiksi Internet tuodaan Edgen avulla operaattoriverkkoon ja tästä Core mahdollistaa Aggregation -verkon avulla palveluiden siirtämisen Access -laitteille.

Pääasiassa nykyaikainen metroverkko toimii MPLS -tekniikan turvin, jonka etuna on al-

haiset viiveet operaattoriverkon sisällä. Käytännössä Core ja Metro -reitittimet ovat näkymättömiä loppukäyttäjälle erilaisten virtuaaliverkkojen ansiosta eli esimerkiksi Internet on vain palvelu muiden joukossa.

2.3.2 Aggregation

Aggregation verkko koostuu L3 tason reitittimistä, joilla pystytään reitittämään erilaisia palveluita asiakasliityntöihin. Metrolaitteessa on useita SFP -moduulipaikkoja, joihin voidaan liittää kuitu- tai kuparimoduuleja. Eri moduulien käyttäminen kiinteiden porttien sijaan antaa Access tai asiakasreitittimelle vapauden olla yhdistettävissä eri liittimillä muuhun verkkoon. Yleisesti Metrot yhdistyvät toisiinsa kuituverkon avulla sekä samaa tapaa käytetään Access laitteiden yhdistämisessä Metroverkkoon. Hyötynä tässä on skaalautuvuus ja siirtomatkat, jotka voivat olla yli 80 kilometriä riippuen SFP -moduulien tehokkuudesta. Huonona puolena voidaan pitää kustannuksia ja kuidun herkkyyttä fyysisille rasituksille.

2.3.3 Access

Näitä asiakasliityntöjä tarjoavia verkkolaitteita kutsutaan yleisesti Access -laitteiksi. Näihin verkkolaitteisiin lasketaan esimerkiksi matkapuhelinverkon tukiasemat, DSL -keskitin sekä L3 tason Ethernet -reitittimet. Asiakasliitynnän tiedonsiirrossa voidaan käyttää mitä tahansa tapaa siirtää tietoa, mutta asiakas näkee käytettävän tekniikan olevan Ethernet. Access -laitteen tehtävänä on mahdollistaa asiakasyhteydessä käytettävän tiedonsiirtoprotokollan siirtämisen Ethernetin yli. Ethernet kuparikaapeloinnin maksimipituuden ja langattomien siirtoteiden rajoitusten takia on ollut järkevää hyödyntää jo olemassa olevia koaksiaalikaapeli ja lankapuhelinverkko kaapelointeja. Yhä useammin uudisrakennusten tekniseen tilaan tuodaan valokuituvalmius ja samalla kuidun omistava operaattori toimittaa kiinteistökytkimen asiakasyhteyksiä varten. Puhtaalla Ethernetillä toteutetussa liitynnässä puhutaan FTTx -yhteydestä, jossa asiakasyhteys tuodaan kuidulla mahdollisimman lähelle asiakkaan verkkolaitteita.

3 TIEDONSIIRTO TIETOKONEISSA

Tämä kappale kertoo kuinka tietotekniset piirit kommunikoivat ja miten olemassa olevia tekniikoita on jatkokehitetty mahdollistamaan tiedonsiirto toisistaan riippumattomien laitteiden välillä. Verkkotekniikoiden kehittäminen on lähtenyt liikkeelle tarpeesta kommunikoida kahden mikropiirin välillä ja tämän tarkoituksena on hyödyntää toisen piirin ominaisuuksia, jolloin kaikkia ei ole tarvinnut lisätä yhdelle sirulle. Piirien välille muodostetaan kommunikointiväylä yhdistämällä yksi tai useampi sähköinen liitäntä mikropiirien välille. Näiden liitäntöjen jännitetasoja muuttamalla muodostetaan eri kombinaatioita, joiden mukaan vastapää reagoi tietyllä tavalla. Mikropiirien monimutkaistuessa mikroprosessoreiksi ja siirtoetäisyyksien kasvaessa on tullut tarve kehittää tiedonsiirtoon keskittyneitä standardeja, joiden avulla eri valmistajien järjestelmiä voidaan yhdistää toisiinsa ongelmitta.

Rinnakkaismuotoinen tiedonsiirto on edelleen yleisesti käytössä kun halutaan nopeaa ja viiveetöntä tiedonsiirtoa samalla piirilevyllä. Suurin ongelma rinnakkaismuotoisuudessa onkin tarve usealle johtimelle, jolloin tällä tavalla toteutetut nopeat yhteydet vaativat paksun kaapelin tiedon siirtämiseen. Siirtyminen sarjamuotoiseen tiedonsiirtoon on mahdollistanut kustannustehokkaan tavan siirtää tietoa pidemmänkin välimatkan päähän. Oheislaitteita yhdistäessä on ennen USB -määritelmän yleistymistä käytetty RS232 -standardia, joka määrittelee elektroniikka- ja signaalitason vaatimukset laitteiden yhteensopivuudelle. Tiedonsiirron perusperiaatteet eivät ole kovinkaan paljoa muuttuneet RS232 -standardin tyylistä lähettää merkki kerrallaan vastaanottajalle ja nykyäänkin USB -tekniikka perustuu samaan periaatteeseen, mutta yhden USB3 paketin datakehykseen mahtuu enintään kilotavu tietoa.[5] Vaikka USB -tekniikka on yleinen tapa yhdistää oheislaitteita tietokoneeseen on sen ongelmana jokaisen laitteen vaatimat ohjainajurit, jonka takia varsinkin RS232 tekniikka on säilynyt laitteiden konsoliliitäntöjen standardina. Olemassa olevien USB-RS232 -adaptereiden avulla pystytään nykyaikaisessakin tietokoneessa hyödyntämään sarjaporttiyhteyttä, vaikkakin sama USB -ohjainajuriongelma hankaloittaa näiden käyttöä eri käyttöjärjestelmissä.

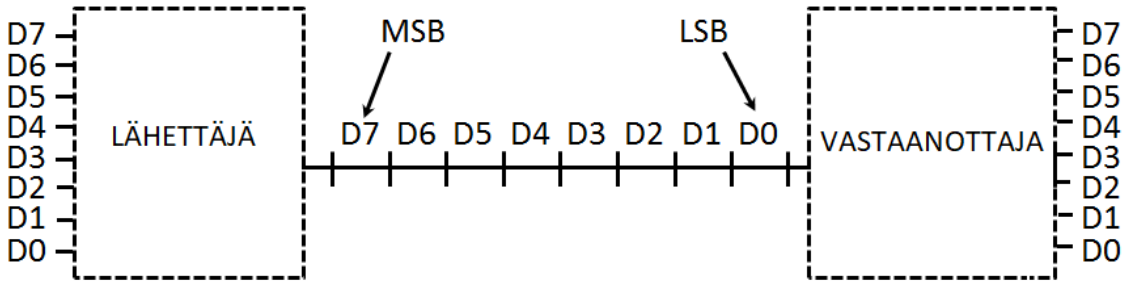
3.1 Rinnakkaismuotoinen tiedonsiirto

Rinnakkaismuotoinen tiedonsiirto on yleistä mikropiirien välillä. Mikropiirien välille muodostetaan useamman johtimen muodostamia tiedonsiirtoyhteyksiä ja näitä kutsutaan väy-

liksi. Väylien avulla piirit jakavat suuren määrän tietoa nopeasti, jonka siirtäminen tahdistetaan esimerkiksi kellosignaalilla. Väylänleveyden määrää kuinka monta bittiä pystytään siirtämään kerralla siirtokaistalla ja yleisesti kaapeleissa käytetään signaalin siirtämiseen yhden tavun kaistanleveyttä johtimien säästämiseksi. Kaistanleveyttä kasvattamalla pystytään helposti siirtämään suuri määrä tietoa, jolloin tahdistuksessa käytettävää taajuutta ei tarvitse nostaa. Oheislaitekytkennöissä rinnakkaismuotoisuutta on käytetty tulostimien yhdistämiseen, joka ennen USB -yhteyksiä oli sarjamuotoisuutta nopeampi tapa tähän tarkoitukseen. IEEE 1284 standardin mukaisessa tulostinliitännässä on 8-bitin eli tavun kaistanleveys, jolloin kaapelin paksuus ja siihen tarvittavan liittimen koko pysyvät järkevänä. Varsinaisten datajohtimien lisäksi tiedonsiirto ohjaamaan tarvitaan myös muita signaaleja, jotka esimerkiksi kertovat vastapuolelle milloin se on valmis vastaanottamaan tai lähettämään tietoa dataväylältä.[6] [7]

3.2 Sarjamuotoinen tiedonsiirto

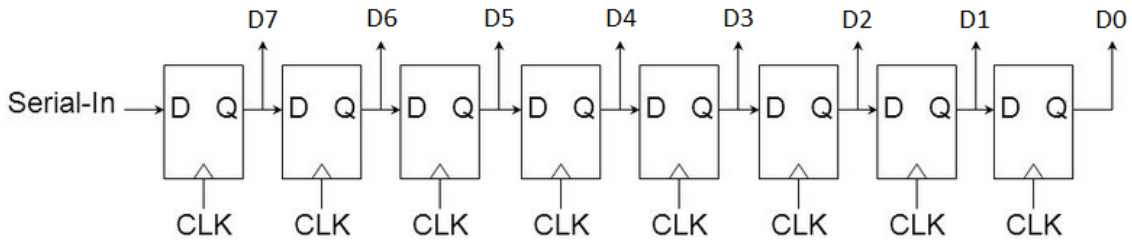
Rinnakkaismuotoisen signaalin muuntaminen sarjamuotoiseksi voidaan toteuttaa siirtorekisterillä, joka lukee tiedon väylästä ja siirtää jokaisen johtimen signaalin tietyssä järjestyksessä yhdelle johtimelle. Kuvasta 3.1 voidaan nähdä miten rinnakkaismuotoinen tieto muutetaan sarjamuotoiseksi lähettämällä Vähiten merkitsevä bitti (LSB) ensimmäisenä ja sitten bitti kerrallaan kunnes viimeisenä Eniten merkitsevä bitti (MSB).



Kuva 3.1. *Yksinkertaistettu muunnos sarjamuotoiselle tiedonsiirrolle.*

Sarjamuotoista tiedonsiirtoa käytetään sellaisissa yhteyksissä missä viive ja kaistanleveys ei ole niin tärkeässä osassa tiedonsiirtoa. Yleinen tapa muuntaa rinnakkaismuotoinen data sarjamuotoon tai toisinpäin on käyttää PISO ja SIPO siirtorekistereitä. Muunnoksista yksinkertaisempi SIPO eli sarjamuotoisesta rinnanmuotoiselle on esitetty kuvassa 3.2. Muunnokseen tarvitaan yksi D-kiikku per ulostuleva bitti ja kiikkuja lisäämällä voidaan muodostaa lähes kuinka leveä väylä tahansa. [8] [7]

Merkkejä siirrettäessä käytetään yleensä kahdeksan bittiä leveää väylää, joka riittää perus ASCII -merkkien siirtämiseen. Jos merkin pituus on vähemmän kuin siirtokaistan leveys, voidaan sen perään lisätä tarvittava määrä nollia täyttämään tyhjä bitit. Käytännön sovelluksessa väylää ohjataan erilaisilla ohjaussignaaleilla ja siirryttäessä sarjamuotoiseen tiedonsiirtoon on nämäkin signaalit otettava jollain tavalla huomioon.



Kuva 3.2. *D-kiikuilla toteutettu 8-bittinen sarja-rinnan muutos.*

Sarjamuotoinen tiedonsiirto on teoriassa mahdollista yhden johtimen avulla, mutta jos laitteilla on erillinen maapiste, on lisättävä toinen johdin määräämään maapisteen laitteiden välille. Tiedon siirtäminen yhdellä johtimella on mahdollista kun siirretään tietoa yhteen suuntaan, mutta kaksisuuntaisessa siirrossa pelkästään samanaikainen lähettäminen aiheuttaa paketin vääristymisen. Lisäksi Käytännössä kaksisuuntainen tiedonsiirto on mahdollista kolmella johtimella, joista kaksi hoitaa tiedonsiirron suuntiinsa ja kolmas määrää jännitteen nollapisteen. [9]

3.3 Asynkroninen tiedonsiirto

Kellolla synkronointi on oleellista ohjelmoidessa yksinkertaisia laitteita, mutta kommunikoidessa oman kellon omaavien laitteiden kanssa voidaan jättää kyseinen signaali lähettämättä, tällä tavalla toteutetun siirtotapaa kutsutaan asynkroniseksi. Samalla kuitenkin häviää tieto loogisien tasojen potentiaaleista, jonka takia ainakin maataso on jaettava laitteiden välillä. Lisäksi historiallisesti on lähettävän pään perustila eli looginen nolla jännitteellinen, jonka ansiosta vastaanottava pää pystyy todentamaan linjan olevan kytketty. Huonona puolena tässä on siirtokaistalla tapahtuva jännitehäviö, jonka takia moderneissa rinnan- ja sarjamuotoisissa ratkaisuissa on maataso nolla volttia.

Kellotahdistuksen puuttuessa on sarjamuotoiselta tiedonsiirron haasteena luotettava tiedon lähetys ja vastaanotto. Asettaessa siirtotavalle omat vaatimuksena puhutaan protokollasta eli säännöstöstä. Vakaa ja virhevapaa tiedonsiirtoprotokolla sisältää tiedon siirtokaistalla käytetystä nopeudesta sekä sillä siirrettävä datakehyksellä on vähintään alku, itse tieto sekä loppu. Valinnaisena lisämääreenä datakehyksellä on datasta laskettu pariteettibitti, jolla karkeasti kerrotaan vastaanottavalle päälle onko loogisien bittien yhteenlaskettu summa pariton vai parillinen, lopputulos määrää hyväksytäänkö paketti ehjänä. Kuvasta 3.3 nähdään sarjaliikenteessä käytetyn paketin kehys, joka sisällöstä riippuen on enintään 12 bittiä pitkä. Yksinkertaistetussa asynkronisessa tiedonsiirrossa alkubitti ilmoittaa vastaanottopäälle informaation lähettämisestä jonka jälkeen seuraa varsinainen merkki binäärimuodossa. Lopuksi tulevat pariteettitarkistus ja lopetusbitit.

Tiedonsiirtonopeus ilmoitetaan bittiä per sekunti, jonka suuruus määräytyy pääasiassa hitaamman laitteen suorituskyvystä. Yleinen tietoliikennelaitteiden konsolikäytössä ole-

Kuva 3.3. *Asynkronisessa ja synkronisessa tiedonsiirrossa käytetyn paketin rakenne. [9]*

va nopeus on 9600 bittiä sekunnissa, joka on ollut kompromissi sen ajan tietokoneiden suorituskyvyn ja tiedonsiirron nopeuden välillä. Asynkronisen tiedonsiirrossa olevien laitteiden nimeämistä on haluttu selkeyttää DTE-DCE -parilla. Isäntäkonetta tarkoittaessa puhutaan lyhenteestä DTE eli Data Terminating Equipment ja oheislaitteesta vastaavasti DCE eli Data Communication Equipment. DTE termiä pääasiassa käytetään laitteista jotka tuottavat varastoivat tai tuottavat dataa käyttäjälle. Tietokoneet, terminaalit ja tulostimet kuuluvat DTE laitteisiin. DCE laitteet mahdollistavat pääsyn järjestelmiin verkon yli. Tähän kategoriaan sisältyvät modeemit ja kytkimien kaltaiset laitteet. [9] [10]

3.4 UART sarjaliikennepiiri

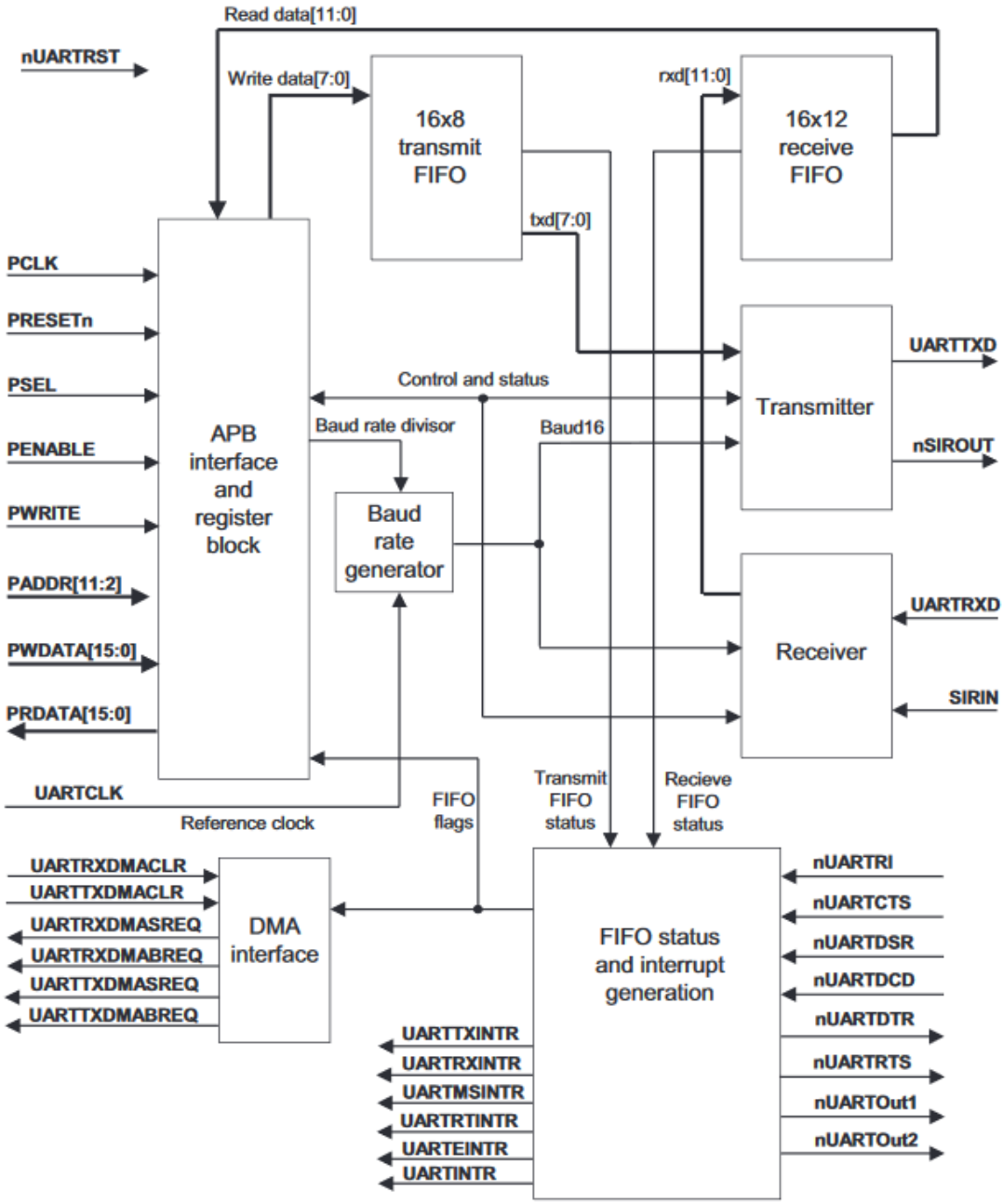
UART on asynkroniseen sarjamuotoiseen tiedonsiirtoon tehty tietoliikennepiiri. Yhden piirin alta löytyy useampi lohko jotka hoitavat molempien suuntien tiedonsiirrossa tarvittavia toimenpiteitä. Nykyaikaisissa prosessoreissa UART on toteutettu piirin sisäisenä moduulina, jolloin säästetään arvokasta pinta-alaa piirilevyltä sekä itse pinnejä mikropiiriltä. Alati pienenevillä mikropiireillä on tärkeää säästää tilaa sen varsinaisille tarpeille, kuten esimerkiksi muistiväylälle tai virransyötölle.

UART on väylä, joka ei itsessään ota kantaa mitkä ovat loogisien tasojen jännitteet, vaan nämä määräytyvät mikropiirin sisäisten ns. TTL jännitteiden mukaan. Nykyaikaisissa vähävirtaisissa SoC -piireissä voi TTL käyttöjännitteet vaihdella 1,8 voltista 5 volttiin, mikä tarkoittaa että samaa jännitettä käytetään signaloidessa toisen piirin kanssa. Tämän takia se ei kelpaa sellaisenaan kahden eri laitteen kanssa signalointiin, jossa pitkä välimatka heikentää signaalia entisestään.

Raspberry Pi tietokoneissa käytetty ARMin BCM283x -sarjan prosessorit käyttävät pääasiallista PL011 UART -liitäntää konsolikäytössä. Raspberry Pi versiosta 3 lähtien on PL011 siirretty Bluetooth piirin käyttöön, mutta konsoliyhteys on edelleen mahdollinen prosessorista löytyvän Mini-UART liitännän kautta. Pääasiallisena erona PL011:een on virheentarkistusbitin puuttuminen sekä sen siirtonopeus on sidottu piirin ytimen nopeuteen.

Kuvassa 3.4 on esitetty PL011 UART moduulin lohkokaavio, josta voidaan tarkastella UARTin toimintaa logiikkatasolla. Oikeassa reunassa näkyvät siirrosta vastaavat lohkot, jotka käytännössä tekevät datamuunnoksen sarjamuotoiseksi ja takaisin. Eroina lohkoilla on että lähetyslohko kehystää saadun 8 bittisen datan kuvan 3.3 muotoon eli kehystävät

datan siirtokaistalle. Vastaanotossa kehys tarkistetaan ja poistetaan kehyksestä. Merkki lähetetään eteenpäin vaikka virheitä olisikin sattunut ja mukaan lisätään 3 bittiä ilmaisemaan mahdollisia virheitä sekä yksi bitti mahdollisesta puskurin täyttymisestä. Merkit varastoidaan 16 merkin kokoisissa FIFO puskureissa, jotka myös tarvittaessa kommunikoivat vuonhallinnasta vastaavan keskeytyslohkon kanssa. Keskeytyslohko kommunikoi vastapään ja muiden lohkojen kanssa milloin tietoa voidaan siirtää. DMA ja APB toimivat suoraan prosessorin kanssa varsinaisina datansiirron ohjaimina lopulle UART -piirille. Viimeisenä tiedonsiirtonopeuden määrää Baud rate generator, joka käytännössä tuottaa kellopulssista siirtonopeuden siirtolohkoille. [7] [11] [12] [13]



Kuva 3.4. *PL011 UART -moduulin lohkokaavio. [13]*

3.5 RS232 standardi

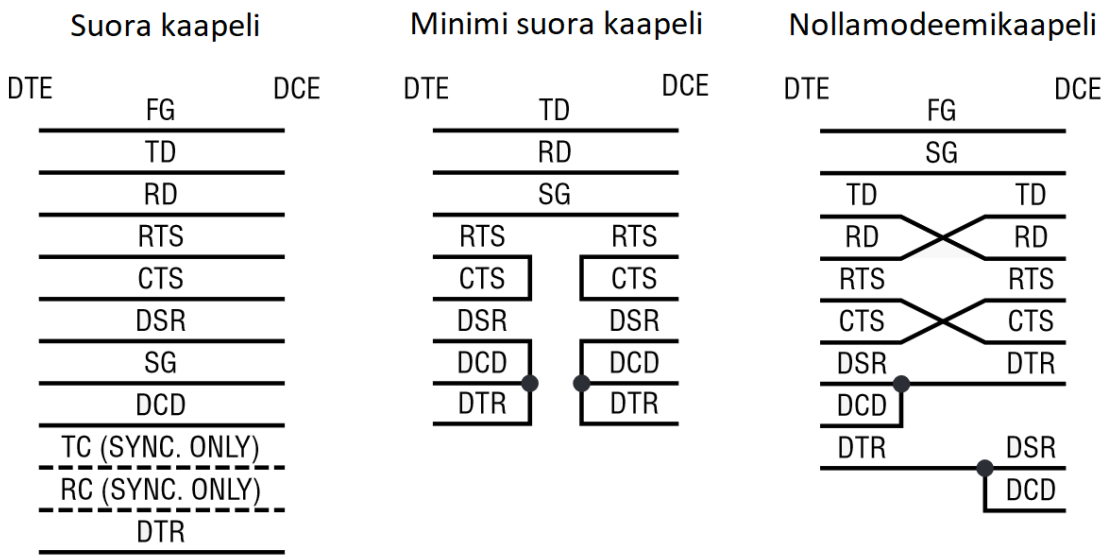
Puhuttaessa RS232 -standardista tarkoitetaan yleisesti verkkolaitteista löytyvästä asynkroniseen tiedonsiirtoon käytetystä sarjaliikenneportista. Standardin kehittäminen alkoi 1960 -luvulla ja sitä on päivitetty ja laajennettu vuosien mittaan eri tarpeiden mukaan. Standardin viimeisin versio kulkee nimellä TIA-232-F, joka määrittelee DTE-DCE parin tiedonsiirtoon tarkoitetut liittimet, siirtokaistan vaatimukset ja signaalitasot. Standardi on ennen USB -tekniikkaa ollut yleisesti käytössä yhdistettäessä oheislaitteita tietokoneeseen ja myöhemmin sitä on myös hyödynnetty kahden tietokoneen lähiverkon muodostamiseen. Suurin ero TTL signaaliin on käyttää loogisena ykköstilana negatiivista jännitettä ja standardi määrittää etteivät avoimen piirin jännitetasot saa ylittää +-25 volttia, mutta vastaanottavan pään on tunnistettava vielä +-5 voltin vaihtelut. Kaapelin pituus voi olla enintään 15 metriä, mutta tarkemmin standardi määrittää siirtokaistan kapasitanssin pienenemisen siirtonopeuden kasvamisen mukaan. Alle 20 Kb/s siirrossa kapasitanssi pitää olla 2500 pF, mutta sen nopeuden kasvaessa 1000 pF. [14]

TIA-232-F -standardissa määritellyt jännitetasot ovat nykyaikaisten tietokoneiden jännitteisiin verrattuna kovin voimakkaita ja ratkaisuna tähän on määritelty TIA-562, joka alentaa avoimen piirin jännitetasot +-13,2 volttiin sekä vastaanottavassa päässä täytyy olla +-3,7 volttia. TIA-562 -standardia tukeva liityntä on kuitenkin yhteensopiva TIA-232-F kanssa, jolloin vastaanottajan on kestettävä alkuperäisen standardin jännitetasot. Muita TIA-232-F -standardia laajentavia ovat esimerkiksi TIA-561, jossa määritellään käytettäväksi 8C8P -liitintä pinnijärjestyksineen sekä TIA-574, jonka avulla taataan yhteensopivuus PC -tietokoneiden DE9 -tietoliikenneportin kanssa. [15]

Standardissa määritellyn DTE-DCE -parin vaihtamat signaalit takaavat onnistuneen tiedonsiirron, mutta ajan myötä tietokoneista on tullut nopeampia mikä on vähentänyt ohjaussignaalien tärkeyttä. TIA-232-F määritteleekin että minimissään 3 -johdinta riittää kahdensuuntaiseen tiedonsiirtoon. Lisäksi on määritelty ns. nollamodeemikaapeli, jota käytetään tapauksessa missä luodaan yhteys kahden DTE -laitteen kanssa. Kuvassa 3.5 nähdään miten eri RS232 -signaalit kytketään sovelluksissa. [16]

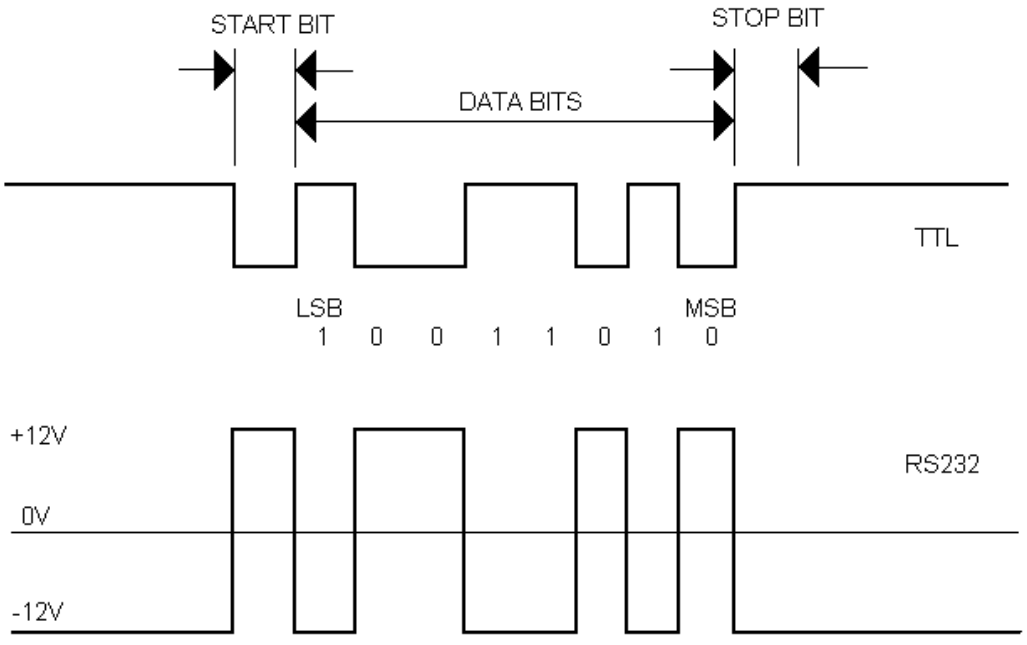
TIA-232-F -standardin yleisyydestä kertoo hyvin se, että puhuttaessa tietoliikenneportista tarkoitetaan DE9 -liitännällistä sarjaliikenneporttia. Nykyajan tietokoneista sarjaliikenneportit ovat jo käytännössä poistuneet, mutta RS232 -yhteyden yksinkertaisuus on säilyttänyt sen yleisen käytön tietoliikennelaitteiden konsoliliitännän standardina.

Kuvassa 3.6 on esitetty TTL ja RS232 -tason signaalien erot. Lepotilassa ja loogisessa ykköstilassa TTL -signaali on jännitteellinen, joka RS232 signaalissa on käänteinen eli negatiivinen. Datasiirron alkaessa jännitetaso vaihtuu päinvastaiseksi, joka tarkoittaa TTL - tilassa maatasoa ja RS232 taas positiivista jännitettä. RS232 -signaalin toiminta perustuukin siihen että aktiivinen yhteys on jatkuvasti jännitteellinen.



Kuva 3.5. *Eri tyyppiset TIA-232-F signaalireitit.*

Kuvan 3.6 signaalissa lähetetään ”Y” -merkki binäärisenä ja tässä on hyvä huomata databittien pituuden olevan 8 bittiä pitkä. Lähetetty merkki on binäärimuodossa oikealta vasemmalle 0101 1001, jolloin vähiten merkitsevä bitti (LSB, Least Significant Bit) lähetetään ensimmäisenä. Sama merkki voidaan esittää heksadesimaalina 0x59 merkinnällä.



Kuva 3.6. *RS232 ja TTL signaalin ”Y” -merkin lähettäminen.*

3.6 RS232 adapterit

DE9 -tietoliikenneportin poistuminen tietokoneista on luonut tarpeen toteuttaa kyseinen liitäntä erilaisilla adaptereilla. Yleisin tapa on hyödyntää tietokoneen USB -liitäntää USB-RS232 adapterilla. Adapterit käytännössä tekevät USB-TTL-RS232 -muunnoksen, eli muuntaa USB liikenteen takaisin sarjamuotoiseksi TTL -tason signaaliksi ja lopuksi RS232 signaaleiksi. Haasteena USB-RS232 adaptereissa on eri valmistajien omat USB -piirit, joiden ohjaamiseen tietokoneessa täytyy olla ajurit asennettuna. Toinen vaihtoehto on käyttää tietokoneen omaa UART yhteensopivaa liitäntää, jos se vain on käyttäjän hyödynnettävissä. Raspberry Pi alustan GPIO -liitännät mahdollistavat prosessorin oman MiniUART -liitännän hyödyntämisen ja siihen voi esimerkiksi kytkeä suoraan MAX3232 -piirin tekemään TTL-RS232 muunnoksen. MAX3232 -piiri toimii kuitenkin pelkkänä TTL-RS232 signaalitasojen muuntimena kahdelle linjatasolle ja normaalisti niillä muunnetaan datasignaalien muunnos. [17]

4 VERKKOLAITTEIDEN KÄYTTÖÖNOTTAMINEN

Tässä osiossa kerron mitä verkkolaitteiden käyttöönottaminen tarkoittaa käytännössä. Perinteinen tapa on muodostaa konsoliyhteys tietokoneen ja verkkolaitteen välille terminaaliohjelmalla ja konsolikaapelilla ja antaa sille yksilöintiin tarvittavat verkkoasetukset. Selvitän myös yleisiä tapoja millä valmistajat ovat mahdollistaneet omien verkkolaitteidensa automaattisen käyttöönoton verkon yli.

Käyttöönottamisella tarkoitan verkkolaitteen asetusten muokkaamista niin että se on valmis toimimaan osana suurempaa verkkoa. Suoraan valmistajan tehtaalta tulevilla verkkolaitteilla on yleensä ohjelmisto asennettuna ja ns. tehdasasetukset määriteltyinä ja nämä mahdollistavat sen helpon käyttöönoton. Ongelmana voi olla näiden oletuksena tulevien sisältämät tietoturvaa heikentävät määrittelyt, kuten esimerkiksi automaattinen verkkoasetusten hakeminen, joka väärin käytettynä voi mahdollistaa ulkopuolisen tahon verkkoon pääsyn. Käyttöönotto on fiksuinta tehdä omassa lähiverkossaan ennen laitteen yhdistämistä osaksi suurempaa kokonaisuutta. Muita yleisiä tietoturvaan vaikuttavia määrittelyitä ovat verkkolaitteiden yksilöintiin vaikuttavat asetukset kuten käyttäjänimi, salasana ja suojaamattomat TELNET ja TFTP -protokollat. Pelkästään jo edellä mainittujen asetusten muokkaamisella tai vaihtoehtoisen tapojen käyttämisellä on suuri vaikutus koko verkon tietoturvaan. Operaattoreiden verkkolaitteet asennetaan ns. teletiloihin, jotka myös osallaan vaikuttavat tietoturvan toteutumiseen. Teletiloina voivat toimia talojen puhelinjakamot, tekniset tilat tai ulos asennettavat lukittavat telekaapit.

Automaattisten käyttöönottojen yleistyminen on vähentänyt perinteisellä konsoliyhteydellä tehtävää konfigurointia, mutta on edelleenkin yleistä että määrittelyt joudutaan tekemään manuaalisesti. Konsoliyhteyden muodostamiseksi juuri asennettuun verkkolaitteeseen asentaja tarvitsee tähän sopivan konsolikaapelin sekä tietokoneeseen asennetun terminaaliohjelman. Konsoliyhteyden muodostamiseksi on tiedettävä verkkolaitteen tehdasasetusten mukainen konsolinopeus sekä kirjautumistunnukset. Verkkolaitteiden ohjelmistona toimivat valmistajien omat verkkokäyttöjärjestelmät, jotka on usein räätälöity laitemallien ominaisuuksien mukaan ja käyttöönottoa tekevä henkilön on tärkeää tietää mallikohtaiset ominaisuudet sen turvalliseen määrittelyyn.

4.1 Konsoliportti ja konsolikaapeli

Suuresta osasta ammattitason tietoliikennelaitteista löytyy konsoliportti, joka mahdollistaa niiden konfiguroinnin ja ylläpidon ilman verkkoyhteyttä. Konsoliporttia voidaan myös hyödyntää järjestelmässä tapahtuvien muutosten seuraamiseen, joka tekee siitä hyvän tavan valvoa verkkolaitteen toimintaa ilman verkkoyhteyttä. Olemassa on myös valmistajakohtaisia ratkaisuja verkkolaitteiden hallitsemiseksi, kuten esimerkiksi Ciscon tapa yhdistää verkkolaite USB -standardilla oheislaitteeksi. Valmistajakohtaisia erikoisratkaisuja en tässä työssä käsittele, koska yleensä samalta laitteelta löytyy RS232 yhteensopiva konsoliportti. Yost on Ciscon määrittelemä pinnijärjestys 8P8C -liittimelle, jossa laitteet yhdistetään toisiinsa käänteisellä kaapelilla (Roll-over cable) eli signaalit näkyvät peilikuvana vastapäässä. Perustana Yostilla on yksinkertaistaa RS232 yhteyksien muodostamista verkkolaitteiden välillä valitsemalla yleisesti Ethernet laitteissa käytetty 8P8C -liitin sekä saavuttaa DTE/DCE pinnijärjestys riippumattomuus. Yost pinnijärjestyksen etuna on myös mahdollisuus hyödyntää TIA-568 standardin suoraa Ethernet kaapelia, jossa samat johdinparit yhdistävät maapinnit sekä datapinnit nollamodeemikaapelin tavalla. Suoran Ethernet kaapelin hyödyntäminen vaatii kuitenkin muunnosadapteria tai DTE/DCE -valitsinta kuten esimerkiksi Nokian SR -reitittimistä löytyy. [18]

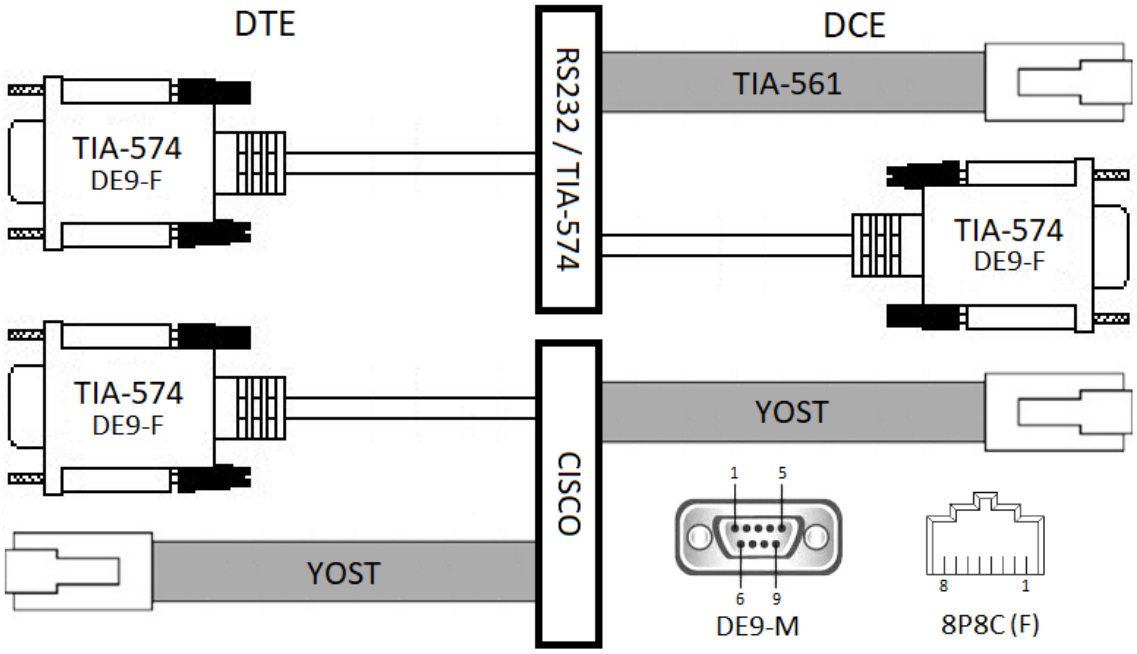
Taulukosta 4.1 nähdään mitkä ovat TIA-232-F -standardin mukaiset pinnijärjestykset DE9 -liittimelle sekä 8P8C liittimen mukaiset järjestykset Yost ja TIA-561 -standardeille.

Taulukko 4.1. *Konsoliporttien pinnijärjestykset standardeineen. [18] [19]*

Signaali	8P8C		RS232 DE9	
	(Yost)	(TIA-561 DCE)	DTE	DCE
RTS	1	8	7	8
DTR	2	3	4	6
TxD	**3**	**6**	**3**	**2**
GND	4	4	5	5
GND	5	4	5	5
RxD	**6**	**5**	**2**	**3**
DSR	7	-	6	4
CTS	8	7	8	7
DCD	-	2	1	1
RI	-	1	9	9

Yleisesti tietoliikennelaitteen konsoliportin tunnistaa sen läheisyyteen painetusta tekstistä CONSOLE tai CRAFT, mutta standardit eivät määrää minkälainen merkinnän pitäisi olla. Kansainvälisempi tapa on värimerkkaus tai kirjoittaa sen läheisyyteen IOI. Merkitseminen on varsinkin Ethernet kytkimissä on suositeltua ettei liitäntää sekoittaisi tavalliseksi Ethernet portiksi. Terminaalin ja verkkolaitteen DE9 -liittimenä käytetään urosporttia, jolloin

konsolikaapelin molemmat päät ovat naarasliittimiä. Kuvassa 4.1 on esitetty yleisimmät kaapelityypit konsolikäytössä.



Kuva 4.1. *Erilaiset konsolikaapeli variaatiot liittimineen.*

Yost -kaapelista on olemassa kaksi erilaista kaapelia; toinen on normaali DE9-8P8C -kaapeli tietokoneen ja verkkolaitteen yhdistämiseen ja toinen 8P8C-8P8C -kaapeli, jota voidaan käyttää DE9 -adapterin avulla konsolikäytössä tai Ciscon oman AUX -liitännän hyödyntämiseen. [19]

4.2 Konsoliyhteys ja ASCII -merkistön käyttö

Konsoliyhteys on yhteydetön yhteys kahden laitteen välillä eli vastapäät lähettävät ja vastaanottavat merkkejä tietämättä toisen olemassaolosta. RS232 ohjaussignaalien käyttäminen konsoliyhteyksistä on lähes poistunut nykyajan tietokoneiden nopeuden ja puskuroinnin ansiosta, mutta toimiva yhteys vaatii yhteisien asetusten määrittämisen yhteydelle. Terminaaliohjelmalle ja verkkolaitteelle täytyy määrittää sama tiedonsiirtonopeus sekä käytetäänkö vuonhallintaa ohjaamaan liikennettä. Yleinen oletukseksi määritelty tiedonsiirtonopeus on 9600 bittiä sekunnissa, joka on osoittautunut riittäväksi tekstipohjaisissa komentoliittymissä. Nopeutta voidaan kuitenkin nostaa portaittain aina 115200 b/s asti, jota nopeammat yhteydet voivat aiheuttaa signaalin vääristymistä TIA-232-F -standardin suositusten ylittyessä. Kommunikoinnissa laitteet lähettävät toisilleen taulukossa 4.2 esitettyä merkistöä binäärisenä, joko ohjaten vastapäätä käyttäjälle näkymättömillä ohjausmerkeillä tai muodostaen ihmisluettavaa tekstiä.

Konsoliyhteydessä lähetetään merkki kerrallaan yhtenä bittijonona paketoituna kuvassa 3.3 esitetyllä tavalla. Normaalisti datakehyksen pituus on 7 bittiä, joka kattaa kaikki käy-

tettävät ASCII -merkit konsoliympäristössä. Perus ASCII sisältää yhteensä 128 eri merkkiä, joista 33 merkkiä on tarkoitettu laitteiston ohjaukseen ja lopuilla voidaan muodostaa mahdolliset verkkokäyttöjärjestelmässä tapahtuvat tulosteet ja komennot englannin kielellä. Laitteistonohjausmerkkien avulla verkkokäyttöjärjestelmät osaavat kertoa konsolipäätteelle kuinka teksti tulisi tulostaa päätteessä. Lisäksi käyttäjän syöttämien ohjausmerkkien avulla voidaan antaa komentoliittymille käskyjä esimerkiksi keskeyttää ajossa olevan käskyn suorittaminen tai tekstin tulostamisen päätteelle.

ASCII perustaulukkoa voidaan pitää tietokonejärjestelmien ohjauksen perusmerkistönä, mutta siihen on tehty useita erilaisia laajennoksia tukemaan esimerkiksi skandinaavisista merkistöä. Unicode laajentaa merkistön 21 bittiseksi, joka riittää kattamaan kaikki maailman kielet ja tilaa jää vielä mahdollisille sisäyksille. Unicoden ensimmäiset 128 merkkiä ovat kuitenkin samat kuin alkuperäisessä ASCII:ssa. Merkkejä voidaan esittää joko 7 bitin binäärimuodossa tai 00-7F hexadesimaalina. Kuvassa 4.2 nähdään 7-bittisen ASCII merkistö ja mitkä ovat niiden hexadesimaaliarvot niitä esittäessä. Hexadesimaalia käytettäessä sen alkuun 0x -merkintä ilmaisemaan sitä ja esimerkiksi merkki Y voidaan esittää 0x59 -koodilla. [20]

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Kuva 4.2. *Konsolikäytössä yleisesti käytetty 7-bittinen ASCII merkistö. [21]*

4.3 Valmistajakohtaiset verkkokäyttöjärjestelmät

Tietoliikennelaitteissa on valmistajakohtaisia käyttöjärjestelmiä ohjaamaan verkkolaitteen elektroniikkaa. Kytkimissä ja reitittimissä käytössä olevaa käyttöjärjestelmää ohjataan omasta tekstipohjaisesta käyttöliittymästä, johon pääsee kiinni verkkolaitteen konsoliportin kautta. Selainpohjaista hallintaliittymää löytyy kyllä useilta valmistajilta, mutta yleensä niihin päästään käsiksi vasta kun laite on esikonfiguroitu tai ne on tarkoitettu pääasiassa kuluttajakäyttöön. Yksinkertaisemmissa tekstipohjaisissa järjestelmissä käskyjen muodostaminen on rakenteellista, eli puumaisesti muodostavat sana sanalta tietyn määrityksen kuten esimerkiksi onko tietty portti sähköisesti auki vai kiinni. Tämän diplomityön kannalta oleellisin verkkokäyttöjärjestelmä on Huawein omissa L3 verkkolaitteissa käyttä-

mä Huawei VRP, jonka käskyrakenne hyvin samankaltainen kuin Cisco Systemsin IOS -verkkokäyttöjärjestelmä. Ylläpidon kannalta rakenteellisissa verkkokäyttöjärjestelmässä on annettujen asetusten välitön käyttöönotto, jonka takia on kehitetty modulaarisia järjestelmiä joissa käskyjä voidaan jakaa lohkoissa suoritettaviksi. Tällaisia ovat esimerkiksi Juniperin Junos ja Ciscon IOS XR -käyttöjärjestelmät.

4.4 Verkkolaitteiden manuaalinen käyttöönotto ja sen hyödyt

Perinteisin tapa käyttöönottaa verkkolaitteita on yhdistää pääte-emulaattorilla varustetulla tietokoneella laitteen konsoliporttiin, määritellä konsoliyhteyden asetukset ja syöttää asetuksia verkkolaitteeseen. Asetukset määräytyvät pitkälti mihin käyttöön verkkolaite tulee sekä kuinka se liitetään osaksi suurempaa verkkotopologiaa. Jos kuitenkin tehdasasetuksissa on se asetettu hakemaan verkkoasetukset DHCP -palvelimelta automaattisesti, voi siihen yhdistää toiseen verkkoon kuuluvalta laitteelta TELNET tai SSH -yhteyden avulla. Kriittisesti tärkeissä verkoissa, kuten esimerkiksi operaattoriverkoissa, on tärkeää estää ulkopuoliset tahojen pääsy muihin verkon laitteisiin. Hyvän tietoturvan lähtökohta on minimoida luvatonta pääsyä ja eristää verkon eri osia toisistaan. Esimerkiksi kuluttajaliityntä voidaan eriyttää omaksi verkokseen Metro Ethernet -tyyppisillä palveluratkaisuilla, joissa kuluttajalle määrätty Access -laitteen portti sisältää vain Internetiin pääsyn mahdollistavan palvelun. Kuvassa 2.2 toteutetun Metro Ethernet verkon Access -laitteet on yhdistetty suoraan Aggregation verkon reitittimiin. Tietoturvan kannalta on tärkeää pitää pääsy tällaiseen verkkoon erittäin rajattuna tarkoittaen uusien laitteiden manuaalista esiohjelmointia. Esiohjelmoinnilla voidaan varmistaa ettei asennettava verkkolaite toimi kuin sille tarkoitetulla tavalla.

4.5 Käyttöönottojen automatisointi

Yleensä valmistajat toimittavat kuluttajakäyttöön tarkoitetut verkkolaitteensa käyttövalmiina eli niistä löytyy valmiina käyttöjärjestelmä ja automaattinen osoitteiden haku. Nämä Kuluttajakäyttöön tarkoitetut laitteet ovat yleensä Access -kiinteistökytkimiin yhdistettäviä asiakasreittimiä, joiden tehtävänä on luoda oma lähiverkko asiakkaan omille tietokoneille. Nämä kuluttajareitittimet oletusmäärityksillä hakevat DHCP -protokollan avulla operaattorin määrittelemästä osoitealueesta ainakin IP -osoitteen, aliverkkopeitteen ja oletusyhdyskäytävän. Nämä tiedot on operaattorin itse määrittelemiä asetuksia DHCP -palvelimeen ja ne jaetaan verkkolaitteelle DHCP:n Option kenttien avulla. Eri tyyppisiä DHCP -paketteja on kahdeksan, joista neljä on tarkoitettu uuden laitteen tuomiseksi verkkoon. Loppuja pakettityypeistä käytetään osoitteiden hallintaan ja viansietoon. Taulukossa 4.2 esitetään mitä viestejä palvelin ja laite lähettävät sekä mitä olennaista sisältöä ne käyttöönoton kannalta sisältävät.

Taulukko 4.2. *Esimerkki DHCP -protokollan jakamista paketeista ja yleisesti käytetyt Optionit käyttöönoton automatisoinnissa. [22] [23] [24]*

DHCP paketti	Lähettäjä	Vastaanottaja	Yleiset Optionit
DHCP Discover	Laite	Broadcast	55*,60,61
DHCP Offer	Palvelin	Broadcast	- (vain verkkoasetukset)
DHCP Request	Laite	Broadcast	55*,60,61
DHCP ACK	Palvelin	Broadcast	66,67,43

* Koottu tieto haluttavista verkkoasetuksista ja muista Optioneista

Kaikki paketit lähetetään MAC broadcastinä verkkoalueen laitteille. Haku alkaa osoitetta hakevan verkkolaitteen DHCP Discover -paketilla, johon DHCP -palvelin vastaa DHCP Offer -paketilla. DHCP Offer sisältää DHCP Discover paketissa pyydetyt verkkoasetukset, muttei käyttöönottoon liittyviä parametreja. Verkkolaite hyväksyy saadut asetukset lähettämällä DHCP Request -paketin, jolloin DHCP -palvelin tietää varata tarjotun verkko-osoitteen laitteelle. Lopuksi palvelin lähettää laitteelle DHCP ACK -paketin varauksen onnistumisesta sekä laitteen pyytämät käyttöönottoon liittyvät asetukset. [22]

Automatisoiduissa käyttöönotoissa käytetyt Option -parametrit vaihtelevat valmistajan sovellusten mukaan. Pääsiassa sovellukset perustuvat taulukossa 4.3 listattujen Optioneiden mukaan, mutta valmistajakohtaisiakin ratkaisujakin löytyy kuten Huawein EasyDeploymentin kirjautusasetusten jakaminen.

Taulukko 4.3. *Yleisesti DHCP -protokollalla jaetuista Option parametreista, joita käytetään käyttöönoton automatisoinnissa. [23] [24]*

Option	Kuvaus	Esimerkki
60	Vendor Class Identifier	"Cisco AP c2800"
61	Client Identifier	Esim. MAC -osoite
66	TFTP Server Name	Palvelimen IP tai nimi
67	Boot File Name	"/kansio/config.py"
43	Vendor Specific Information	5A1N;B2;K4;I172.19.45.222;J80

DHCP -palvelimen ylläpitäjä voi määritellä taulukossa 4.2 näkyvän DHCP ACK -paketin Option 43 -parametriin tietoja, jotka on tarkoitettu tietyn valmistajan laitteille. Esimerkiksi jos Ciscon verkkolaitteita hallitaan Ciscon DNA Center -ohjelmistoa, voidaan Option 43 asetuksella antaa Ciscon palvelinasetukset, omat verkkoasetukset ja esimerkiksi määrätä laitteen virtuaaliverkon (VLAN). Yleisempiä tapoja jakaa palvelintietoja ovat Option 66 ja 67, jotka sisältävät tiedon verkosta löytyvistä paikoista mistä konfiguraatiotiedoston löytää. Haittapuolena Optioiden 66 ja 67 käyttämisessä on niiden mainostus kaikille verkossa niitä pyytäville laitteille. ZTP eli Zero Touch Provisioning termillä markkinoidaan Option 66 ja 67 perustuvia ratkaisuja, joissa hallintajärjestelmää ei hyödynnetä. [23] [24]

Ohjelma 4.1. *Esimerkki Ciscon automatisointi scriptin toteutuksesta. [25]*

```
1 import cli
2 cli.configurep(["ip␣addr␣1.1.1.2␣255.255.0.0","no␣shut","end"])
3 cli.configurep(["ip␣default-gateway␣1.1.1.1", "end"])
4 cli.configurep(["username␣admin␣privilege␣15␣secret␣0␣salasana"])
```

4.5.1 Cisco ZTP ja Plug and Play Provisioning

Cisco tarjoaa verkkolaitteidensa käyttöönottoon kahta erilaista automatisoitua ratkaisua. Ciscon ZTP -ratkaisussa laitteelle määritellään Option 67 mukaisesti konfiguraatiotiedosto, joka voi olla esimerkiksi Python -skripta määrittelemään verkkolaitteen asetukset. Ohjelmassa 4.1 on esimerkki miten verkkoasetuksia määrittävät komennot muodostetaan Ciscon skriptissä. [25]

Ciscon DNA Center toimii Ciscon verkkolaitteiden hallintajärjestelmänä ja sen automatisoitua käyttöönotosta puhutaan nimellä Plug and Play Provisioning. Option 43 määrittelyn mukaan voidaan asettaa Ciscon määrittelemät parametrit ohjaamaan laite yhdistämään DNA Centeriin ohjelmoitavaksi. [26] Taulukossa 4.3 esitettyyn Option 43 -kentän sisältö on tapa miten Ciscon verkkolaitteet saavat DNA Centerin osoitteen ja mitä protokollaa pitää käyttää. [25]

Taulukko 4.4. *Esimerkki Ciscon DNA Center -palvelimen osoitteen antamisesta DHCP Option 43 -avulla [27]*

Tieto	Selitys
5A1N	Cisco PnP, versio 1, ei debug
B2	IPv4 -osoite
K4	HTTP -protokolla
I172.19.45.222	I + Palvelimen osoite
J80	J + Portti mihin yhdistää

4.5.2 Huawein AutoConfig ja EasyDeploy

Huaweilla on pääasiassa kaksi erilaista toteutusta automatisoida verkkolaitteiden käyttöönottoja. ZTP -ratkaisun tyyppinen DHCP -palvelimen Option -parametreihin tukeutuva Auto-Config -ratkaisu, jonka avulla kerrotaan mistä tiedostopalvelin löytyy ja miten sieltä voi ladata tiedostoja. EasyDeploy on pitkälti samanlainen kuin Auto-Config, mutta verkkoon on samalla lisätty kytkin, joka hoitaa samoja tehtäviä kuin Ciscon DNA Center. ZTP -ratkaisussa on erikoista mahdollistaa nopeampien SFTP/FTP -protokollien käyttämisen tiedostojen jakamisessa ja Huawei kutsuu tätä ratkaisua Auto-Config -termillä. Perinteisesti ZTP jakaa Option 66 avulla TFTP -palvelimen osoitteen, mutta kun käytetään turval-

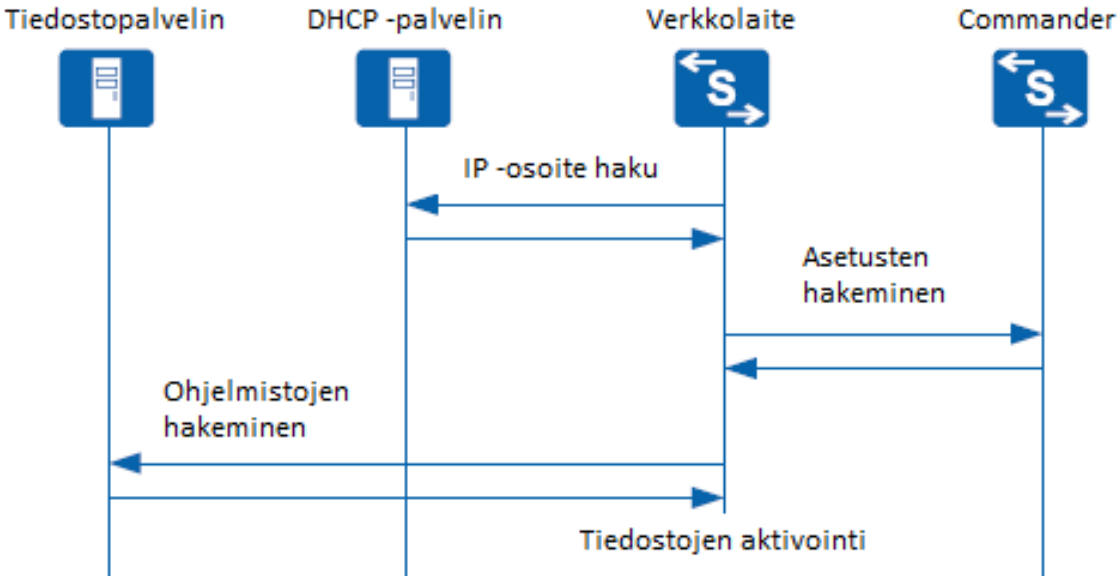
lisempia ja nopeampia SFTP/FTP -palvelimia, tarvitsee verkkolaite tiedon kuinka kirjautua palvelimelle. Taulukossa 4.5 on listattuna mitä Option -kenttiä Huawein Auto-Config tapa käyttää tiedostopalvelimien tietojen jakamiseen. Erikoisen Auto-Config ratkaisusta tekee se, että osa siinä määritellyistä Option -kentistä on DHCP -standardissa varattu muuhun käyttöön.

Taulukko 4.5. *Huawein AutoConfig käyttöönotossa käytetyt DHCP Option -parametrit. [28]*

Option	Kuvaus	Esimerkki
141	SFTP/FTP -palvelimen käyttäjätunnus	admin
142	SFTP/FTP -palvelimen salasana	salasana
143	FTP -palvelimen IP -osoite	ipaddr=10.10.10.1
149	SFTP -palvelimen IP -osoite ja portti	ipaddr=10.10.10.1;port=22
150	TFTP -palvelimen IP -osoite	ipaddr=10.10.10.1
145	Muut jaettavat tiedostot	vrpfile=V200R011C10SPC100.cc
146	Tiedostojen aktivointi	Ajastus ja muut toiminteet
147	Palvelin authentikointi	Määrätty laitetunniste

EasyDeploy perustuu ns. Commander -kytkimeen, jonka avulla hallitaan kaikkia muita verkon kytkimiä. Käyttöönotoissa tälle kytkimelle voidaan määritellä verkkolaitteen tyypin tai MAC -rautaosoitteiden mukaan ohjelmistot sekä verkkoasetukset. Mahdollisuuksien mukaan samainen kytkin voi toimia myös tiedostopalvelimena, jolloin erillistä palvelinta ei välttämättä tarvitse määritellä. Jokaisesta verkkoon lisättävästä kytkimestä lisätään asetukset sisältävä rivi Commanderilla sijaitsevaan lswnet.cfg -nimiseen asetustiedostoon. Rivi koostuu MAC -osoitteesta, laitekonfiguraatiosta sekä mitkä ohjelmistot verkkolaite lataa tarvittaessa. Kuvassa 4.3 on kuvattu Huawei EasyDeployn eri vaiheet kun asetusten jakamiseen käytetään Commander -kytkintä. [29]

Tietoturvaa ajatellen asetustiedostot yksilöidään MAC -osoitteilla, jolloin vain oikea verkkolaite voi saada omat määrittelynsä Commanderilta. Tämän ansiosta muut kuin erikseen sallitut MAC -osoitteet eivät pääse tiedostopalvelimeen käsiksi, sekä kun kaikki laitteet on käyttöönotetut, voidaan Commander ja mahdolliset palvelimet poistaa verkosta. EasyDeploy voidaan myös määritellä toimimaan samalla tavalla kuin Auto-Config, jolloin Commander toimii vain ylläpidollisissa tehtävissä.



Kuva 4.3. *Huawei EasyDeploy automatisoidun käyttöönoton eri vaiheet kun Commander -kytkin toimii asetusten jakajana. [29]*

5 ESIOHJELMOIJAN SUUNNITTELU

Tässä kappaleessa käydään läpi rakennettavalta laitteelta vaadittavat rakenteelliset, tekniset ja ohjelmistolliset ominaisuudet sekä esitellään mitä komponentteja ja ohjelmistoja päädyttiin käyttämään. Tarkoituksena on ideoida miten esiohjelmoijan lopullinen versio tulisi toimimaan ja sen pohjalta luodaan prototyyppi, jossa testattaisiin tämän perustoimintoja. Prototyyppi tulee olemaankin konseptitoteutus, jossa todistetaan että esiohjelmoija on mahdollista toteuttaa.

5.1 Yleiset vaatimukset

Esiohjelmoijasta on tarkoitus tulla työväline asentajan päivittäiseen käyttöön. Käyttö tulisi olla mahdollisimman yksinkertaista, jolloin asentajan lyhyt perehdyttäminen laitteen käyttöön riittäisi. Esiohjelmoijan on tarkoitus olla itsenäinen laite, joka ei tarvitse internetyhteyttä toimiakseen. Rakenteellisesti laite sisältää DE9-M -portilla toteutetun RS232 -yhteensopivasta konsoliliitynnän, kiinteän virtakaapelin, laitteen toimintaa ilmaisemasta käyttöliittymästä sekä yksilöidystä kotelosta. Lisäksi laitteesta on löydyttävä USB -portteja esimerkiksi USB-RS232 -adapteria varten. Vaatimuksia voidaan verrata muihin mukana kuljetettaviin mittalaitteisiin, joista poikkeuksena esiohjelmoija olisi patteriton eli tarvitsisi erillisen jännitesyötön. Komponenttien hyvä ja yleinen saatavuus on tärkeää esiohjelmoijan elinkaaren ja huollettavuuden kannalta, tärkeää onkin että laitteen jonkin osan rikkoutuessa voi asentaja hankkia ne lähimmästä tietotekniikkaa myyvästä liikkeestä ja niiden vaihtaminen on helppoa. Asentajan tehtäväksi jää kytkeä esiohjelmoijaan virta sekä kytkeä konsolikaapeli sen ja verkkolaitteen välille. Normaaliin toimintaan ei tarvitse vaikuttaa, vaan esiohjelmoijan toiminta olisi täysin automaattista. Esiohjelmoijan ohjelman toimintaa pystyisi seuraavaan laitteesta löytyvien merkkivalojen tai näytön avulla. Ohjelmallisesti esiohjelmoija pystyy tunnistamaan tuetut laitteet ja niiden ohjelmistoversiot.

Prototyyppivaiheessa laitteelle tehdään tuki Huawei S -sarjan verkkolaitteissa käytetylle Huawei VRP -verkkokäyttöjärjestelmälle. Prototyypin käyttöliittymää ohjataan tietokoneen terminaaliohjelman avulla ja siihen luotavan ohjelman on pystyttävä ainakin verkkolaitteen konfigurointiin, konfiguroinnin tarkistukseen ja nollaamaan laite tehdasasetuksille.

5.1.1 Rakenteelliset ominaisuudet

Asentajan mukana kulkevat elektroniset laitteet altistuvat usein vaihteleville olosuhteille, jolloin niiden kestävyydeltä vaaditaan enemmän kuin kuluttajalaitteilta. Ammattitason mittaus ja testauslaitteet ovat rakenteellisesti vedenkestäviä sekä ne sietävät rankempaakin fyysistä käyttöä. Koteloinnin ja siihen tuettujen liittimien kestävyys ratkaisevat paljon, vaikka laite olisikin sisältä kuluttajatason elektroniikkaa. Heikoksi kohdaksi kestävällekin laitteelle muodostuu sen liittimiin kohdistuva toistuva mekaaninen rasitus, jotka kuluvat käytön seurauksena ja joudutaan tarvittaessa vaihtamaan. Koteloinnin on tarkoitus suojata toiminnan kannalta tärkeitä komponentteja mekaanisesti, sekä mahdollisuuksien mukaan pölyltä ja kosteudelta. Laitteen portteja ei suoranaisesti käytetä, jonka myötä käyttäjä voi liittää esiohjelmoijaan vain sen toiminnan kannalta välttämättömiä kaapeleita. Liittimien suojaamisen kannalta on myös mahdollista tuoda koteloinnin läpi ns. kiinteitä kaapeleita ja tässä tapauksessa niille toteutetaan vedonpoisto. Kotelointi on tarkoitus toteuttaa 3D -tulostuksella ja tämä mahdollistaa liittimen tukemisen koteloon ja vedonpoistojen toteuttamisen kaapeleille. Kotelointi estää käyttäjää pääsemästä käsiksi esiohjelmoijaan, mutta sen avaaminen on mahdollista huollon mahdollistamiseksi. 3D -tulostuksen avulla voidaan vapaasti vaikuttaa koteloinnin fyysisiin mittoihin ja vedonpoistot voidaan toteuttaa suoraan rakenteisiin. Aukkojen ja paljaiden liittimien määrän minimoimisella halutaan vähentää kosteuden sekä pölyn pääsemistä koteloinnin sisälle.

5.1.2 Ohjelmalliset ja muut tekniset ominaisuudet

Esiohjelmoijan ohjelmiston on tarkoitus olla laajalti tuettu ja pitkälti avointa lähdekoodia. Ohjelmisto on oltava helposti päivitettävissä siirrettävän median avulla, jolloin uusia laitteita pystyy luomaan kokoamalla uuden esiohjelmoijan vastaavista komponenteista sekä kopioimalla ohjelmiston kokonaisuudessaan toiselle muistikortille. Käyttöjärjestelmää ja siihen asennettavia lisäosia ei tarvitse päivittää, jolloin esiohjelmoijaa ei tarvitse jatkuvasti ylläpitää eikä sen normaali toiminta vaaraudu siitä riippumattomien osien muuttuessa. Esiohjelmoijaan asennetaan valmiiksi mahdollisimman monen USB-RS232 -adapterin ajurit, joka mahdollistaa sen vaihtamisen tarvittaessa ilman ohjelmistoon koskemista. Tuki eri valmistajien adaptereille lisätään sitä mukaan kun on tarvetta. Esiohjelmoijan on myös kulutettava vähän virtaa ja sen syöttäminen tapahtuu USB -portin avulla, joko verkkolaitteesta löytyvästä USB -portista tai asentajan mukana kulkevasta USB -virtalähteestä. Jatkokehityksessä voidaan virrankulutusta miettiä uudestaan, varsinkin jos se vaikuttaa esiohjelmoijan käytettävyyteen.

Prototyyppi sisältää oman kiinteän RS232 -portin, jota ei tarvitse USB -porttia vaan se toteutetaan laitteen UART -liitynnän kautta.

5.2 Kehitysalustan valinta

Alusta alkaen esiohjelmoijan alustaksi valittiin Raspberry Pi, koska siitä löytyy kaikki tarvittavat liitännät pienessä koossa. Ohjelmistollisesti Raspberry Pi sisältää laajan tuen Linuxille ja siitä löytyy tuki yleisimmille ohjelmistokielille sekä valmiit kehitystyökalut niille. Vaihtoehtoisena alustana harkittiin myös Arduino -pohjaista alustaa, jonka etuna Rasberry Pi:hin on sen alhainen virrankulutus ja hinta, mutta jatkokehityksen kannalta alustan rajoitteet liittimien ja ohjelmistojen kanssa olisivat tulleet nopeasti vastaan. Raspberry Pi ja Arduino alustoilla on yhteistä GPIO -pinnit, joita voidaan käyttää rinnan ja sarjamuotoisen datan siirtoon sekä ne tarjoavat 5 ja 3,3 Voltin jännitepinnit ulkoisille laitteille. Valintaan vaikutti myös saatavilla olevan dokumentoinnin laajuus sekä alustan yleinen saatavuus ja luotettavuus. Raspberry Pi:n käyttäminen prototyypin alustana olikin järkevä ratkaisu, koska se on yleisesti saatavilla lähes jokaisesta tietotekniikkaan perehtyneestä liikkeestä ja sen toimintakuntoon saattamiseen tarvitsee vain kopioida ohjelmisto SD -muistikortille. Optiona uudemmissa Raspberry Pi -alustoissa on mahdollisuus ladata käyttöjärjestelmä USB -muistilta, mutta ominaisuus pitää erikseen aktivoida alustan sisäisestä muistista. Alustasta on kehitetty eri tarkoituksiin tarkoitettuja versioita, joista yleisimmät ovat Model A ja Model B -mallit. Näiden mallien pääasialliset erot ovat liittimien määrissä, joista A -versiossa on vähemmän USB -portteja sekä Ethernet -liitin toimii. Esiohjelmoijan ja prototyypin kannalta on kuitenkin riittävää, että alustasta löytyy ainakin yksi USB -portti ja GPIO -pinnit. Raspberry Pistä on myös olemassa normaalimalleja karsitumpi ja halvempi Zero -versio, jossa USB-A -liittimet on korvattu yhdellä Micro-USB -liittimellä. Esiohjelmoijan toteuttaminen Zero -alustalle vaatisi Micro-USB -adapterikaapelin käyttämisen USB-RS232 -adapterille sekä GPIO -pinnien juottamista alustalle ja etuna sen käyttämisessä olisi alhaisempi virrankulutus. Raspberry Pi ei oletuksena ole mitenkään koteloitu, vaan kaikki komponentit ja liittimet on sijoitettu suoraan yhdelle piirilevylle ja ovat täten alttiina ulkoisille haitoille. Saatavilla olevat koteloinnit on yleensä toteutettu niin että kaikkia liittimiä voidaan hyödyntää käyttäjän toimesta ja niiden tarkoituksena on suojata alustaa pahemmilta vaurioilta.

Prototyypissä hyödynnetään alustan GPIO -pinnejä, joiden kautta voidaan hyödyntää alustan UART -liitäntää RS232 -portin toteuttamisessa sekä kytkeä merkkivaloja ilmaisemaan ohjelman eri suoritusvaiheita. Prototyypin ominaisuuksien kannalta ei ole väliä mitä versiota alustasta käytetään ja ohjelmistoa aloitettiin kehittämään Raspberry Pi 2 Model B+ -alustalla.

5.3 Muiden osien valinnat

Esiohjelmoijan komponentit voidaan jakaa kahteen eri ryhmään, Raspberry Pi:n toiminnan kannalta pakolliset peruskomponentit sekä esiohjelmoijan toimintaan kuuluvat komponentit. Pakollisista komponenteista itse alustan lisäksi voidaan laskea Micro-USB -

kaapeli virransyöttöön ja verkkovirta-adapteri. Virransyöttöön voidaan myös käyttää mitä tahansa Micro-USB -liittimellistä -kaapelia, jolloin virta voidaan ottaa mistä tahansa USB-A -liittimestä. Suositus on kuitenkin käyttää ulkoista virtalähdettä, mutta virtaa voi kuitenkin ottaa verkkolaitteen USB -portista mahdollisuuksien mukaan. Ongelmana on ettei verkkolaitteiden USB -porttia ole tarkoitettu virransyöttöön kuin korkeintaan USB -massamuisteille. Raspberry Pi:n tehontarve vaihtelee paljon mallin mukaan ja tarpeeseen vaikuttaa paljon se mitä oheislaitteita siihen kytketään. HDMI -näyttöportin sammuttaminen sekä oheislaitteiden irrottamisella saadaan kaikkien mallien tehonkulutus tiputettua huomattavasti alle 2,5 Wattiin, jonka USB 2.0 -standardin mukainen liitin pitää pystyä antamaan jatkuvana kuormana. Prototyypin virrantarve voidaan arvioida olevan alhainen, koska siinä ei ole tarvetta käyttää HDMI -porttia ja useampaa USB -oheislaitteita. Tarkoituksena olisikin että prototyyppi toimisi tarpeen vaatiessa verkkolaitteiden USB -liittimien tarjoamalla virralla.

Esiohjelmoijan käyttöön tarvittavat komponentit ovat RS232-TTL -adapteri, verkkolaitteen konsoliportin pinnijärjestyksen kanssa yhteensopiva konsolikaapeli ja GPIO -pinneihin yhdistettävästä RGB LED -moduulista. RS232-TTL - adapteria käytetään muuttamaan Raspberry Pi:n GPIO -pinnien TTL -jännitetasot TIA-232 standardin mukaiseksi. Adapterina käytetään MAX3232 -piirillä varustettua piirilevyä, josta löytyy DE9 -portti konsolikaapelille.

Visuaalisena käyttöliittymänä päädyttiin käyttämään itse rakennettua kahden RGB LEDin valoyhdistelmää, jotka esittävät yhdistelmillään suoritettavan ohjelman silloisen tilan. LEDien yhdistämisessä täytyy huomioida GPIO -pinnien antama 3,3 Voltin jännite ja vaikka LED kestäisikin kyseisen jännitteen on suositeltavaa käyttää etuvastusta suojaamaan mikropiiriä virtapiikeiltä. Nämä kaksi ratkaisua eivät suoranaisesti täytä esiohjelmoijan komponenttien saatavuus vaatimuksia, vaan ovatkin prototyypin kehitystä helpottavia ratkaisuja.

Jatkokehityksen yhtenä ideana on hyödyntää Raspberry Pi:n DSI -liitintä, johon voidaan yhdistää AMOLED näyttö esittämään tarpeelliset tiedot asentajalle. Näyttö voisi esittää esiohjelmoinnin jälkeen verkkolaitteeseen määritellyt verkkoasetukset ja mahdolliset vikatapaukset. Toinen tavoite on lisätä tuki RS232-USB adaptereille, jolloin erillistä RS232-TTL -adapteria ei enää tarvittaisi. Zero -mallia käytettäessä ei kuitenkaan ole mahdollista käyttää DSI -liittimeen yhdistettävää näytöä.

5.4 Ohjelmistojen valinta

Tässä kappaleessa selvitän mitä Linux -versiota päädyin käyttämään ja mitä ohjelmistoja siihen tarvitsee lisätä esiohjelmoijan toteuttamiseen.

5.4.1 Käyttöjärjestelmä

Raspberry Pi:n asennettiin työpöydätön versio Raspbian -käyttöjärjestelmästä. Raspbian on Raspberry Pi:n normaalikäyttöön suositeltu Linux jakelu ja se sisältää ohjelmistokomponentteja alustan ominaisuuksen hyödyntämiseksi. Sama versiota jakelusta voidaan käyttää kaikissa Raspberry Pi -alustoissa, joka tarkoittaa esiohjelmoijan elinkaaren kannalta sitä ettei se ole riippuvainen tietystä alustasta. Tarkemmin käyttöön valittiin Rapsbian Lite -versio jakelusta, josta on poistettu graafinen käyttöliittymä kokonaan.

5.4.2 Ohjelmointi

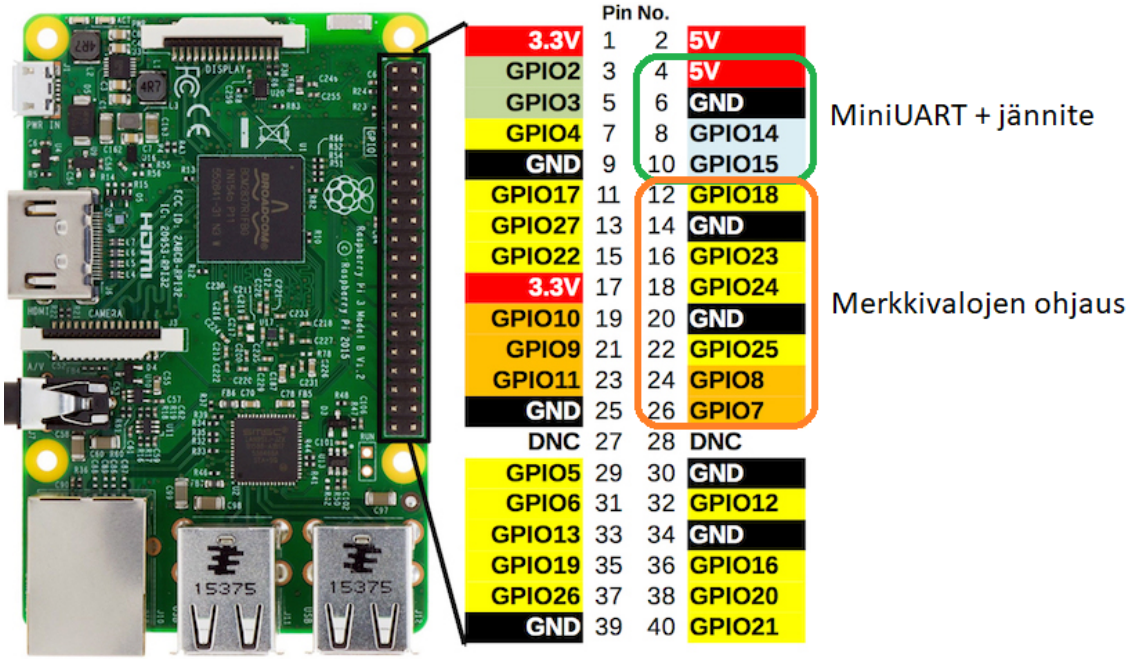
Ohjelmointikieleksi valittiin Python, joka johtui pääosin allekirjoittaneen mieltymyksistä. Päätökseen vaikutti myös käytettävät moduulit, joiden avulla pystytään hyödyntämään Raspberry Pi:n UART -moduulia konsoliyhteyteen. Ohjelmointiin käytettiin Python 2.7.13 versiota ja siihen lisättiin PySerial -moduuli. PySerial mahdollistaa sarjaporttien tuomisen Pythoniin muuttujana sekä mahdollistaa tulosteiden lähettämisen ja vastaanottamisen ohjelmassa. PySerial tukee myös RS232 vuonhallintaa, mutta tässä työssä kaikki kommunikointi verkkolaitteen toimii dataväyliä pitkin.

6 PROTOTYYPIN TOTEUTUS

Tässä kappaleessa kerrotaan mitä ominaisuuksia esiohjelmoijan suunnitelman pohjalta luotuun prototyyppiin toteutettiin. Tarkoituksena oli luoda demolaite, jolla voidaan testata yhden verkkolaitevalmistajan käyttöjärjestelmän ohjelmointia sarjaportin kautta sekä testata Raspberry Pi alustaa ja sen käyttöjärjestelmän soveltuvuutta esiohjelmoijan pohjaksi. Aluksi tehtiin tuki Huawei VRP -verkkokäyttöjärjestelmälle, jonka käskyjen yksinkertaisuus ja samankaltaisuus muiden käyttöjärjestelmien kanssa mahdollistaa toteutetun ohjelman soveltamisen muihin laitteisiin. Prototyypissä käytetään Rapsberry Pi Starter Kitin mukana tullutta koteloa, jonka tarkoituksena on suojata laitetta kolhuilta sekä mahdollistaa kaikkien liittimen hyödyntämisen.

6.1 Fyysinen kokoaminen

Kuvassa 6.1 on esitetty Raspberry Pi -alustan 40-pinninen GPIO -liitin ja mitä pinnejä siitä hyödynnetään lisälaitteita varten.



Kuva 6.1. *Raspberry Pi -alustan GPIO -liitäntä sekä prototyypissä hyödynnettävät pinnit sarjaportille ja merkkivaloille. [30]*

GPIO -pinnirivastoon liitettäville laitteille on mahdollista syöttää käyttöjännite 5 tai 3,3 voltin pinneistä. MAX3232 -piiriä käyttävän RS232 -moduulin käyttäminen on mahdollista molemmilla jännitteillä, mutta koska 5 voltin jännitepinnin sijoittelu oli parempi päädyttiin sitä käyttämään kytkennässä. Lisäksi valintaan vaikutti myös alustan kyky syöttää enintään 50 mA teho 3,3 voltin jännitelinjasta. Sama rajoite on voimassa kun käytetään ohjelmoitavia GPIO -pinnejä, mutta niiden teho on riittävä kunhan niitä käytetään laitteiston ohjaukseen. Mahdolliseen lisälaitteen virransyötölle kannattaakin käyttää 5 voltin pinnejä, koska niiden virransyöttö on suoraan yhdistetty samaan linjaan USB -porttien kanssa. Ulkoisen RS232 -moduulin yhdistämiseen käytetään kuvassa 6.1 esitettyjä vierekkäisiä 4,6,8 ja 10 pinnejä, jolloin oheislaite saa käyttöjännitteen, maatason sekä varsinaiseen tiedonsiirtoon käytettävät signaalit. Merkkivalot yhdistettiin suoraan RS232 -moduulin viereen pinneihin 12-26, jotka sovivasti tarjoavat kahdelle RGB -ledille kuusi kappaletta ohjelmoitavaa IO -pinniä ja kaksi maatasoa. Etuvastukseksi valittiin sopivan kokoinen heti saatavilla oleva vastus, joka on tarkoitus vaihtaa jos varsinkin suuremman jännitteen tarvitsevan sinisen ledin kirkkaus ei ole riittävä tähän käyttötarkoitukseen. [31] [32]

6.2 Ohjelmalliset muutokset

Raspberry Pin käyttäminen esiohjelmoijana tarvitsee muutamia muutoksia käynnistyksen aikaisiin komentoihin. Toiminnan kannalta on tärkeää alustaa GPIO -pinnit toimimaan liitettävien lisälaitteiden mukaan Oletuksena Raspberry Pi:n MiniUART -liitäntä toimii verkkolaitteiden tavoin konsoliporttina, jolloin siihen pystyy yhdistämään konsolikaapelilla tietokoneen sarjaportin kautta kuten muihin DCE -verkkolaitteisiin. Konsoliliitännän muuttaminen normaaliksi sarjaliikenneportiksi onnistuu Rasberry Pi -alustalle tehdyn konsolipohjaisen raspi-config -ohjelman kautta tai suoraan /boot/cmdline.txt -tiedostoa muokkaamalla. Raspi-config -ohjelma on tarkoitettu helpottamaan alustan ylläpitoa ja sillä tässä tarkoituksessa muokataan Linux Kernelin käynnistyksessä suorittamaa cmdline.txt -tekstitiedostoa. Esimerkiksi konsoliasetuksen muuttamiseen tarvitsee lisätä console=serial0,115200 -rivi tiedoston loppuun, joka alustaa UART pinneihin sarjaporttiasetukset ja asettaa sille 115200 b/s nopeuden. Muistikortilla sijaitseva /boot/ -kansio on käytännössä oma levyosionsa, josta Raspberry Pi lataa laitteistoon liittyvät rautatason asetukset alustan käynnistyksessä. Käyttöjärjestelmän käynnistäminen USB -muistilta vaihtelee alustan version mukaan, jonka takia on järkevintä käynnistää se suoraan muistikortilta. Esiohjelmoijassa suoritettava ohjelmisto voidaan kuitenkin tallentaa USB -muistille, jolloin käyttöjärjestelmä sijaitsee eri paikassa. Tarkoituksena on että käyttöjärjestelmän sisältämä muistikortti toimii kaikilla Raspberry Pi -alusta versioilla ja valmiista kortista otetaan levykuva, jota voidaan kloonata useammalle muistikortille. Käytännössä kaikkia myynnissä olevia muistikortteja ei voida käyttää Raspberry Pi alustan käynnistämiseen, jolloin sopivan kortin löytäminen jää käyttäjän vastuulle. Suoritettavan ohjelman automaattinen käynnistäminen on järkevin toteuttaa lisäämällä Python ohjelman käynnistyskomento

/etc/rc.local -tiedostoon. Tällä tavalla ohjelma käynnistyy suoraan kun alusta käynnistetään sekä sen suorittamiseksi ei tarvitse kirjautua käyttäjätilille. Python komennon perään kannattaa lisätä & -merkki, joka suorittaa ohjelman erillisenä prosessina ilman että käyttöjärjestelmän suoritus pysähtyy sen suorittamisessa GPIO -pinnien käyttämiseksi tarvitsee tietää miten yksittäiset pinnit on oletuksena määritelty. Osaan pinneistä on rautatasolla määritelty tiettyjä tehtäviä, kuten esimerkiksi UART ja SPI -liitynnät, mutta niiden toiminnan voi uudelleen määritellä ohjelmistosta käsin. Esimerkiksi merkkivalojen käyttämät GPIO 8 ja 7 -pinnit on oletuksena määritelty SPI -liitynnän hallussa jännitteellisenä, jolloin niiden tila täytyy muuttaa ennen kuin niihin kytkettyjä merkkivaloa voidaan käyttää. Ohjelmistollisesti manuaalisesti ohjatuille GPIO -pinneille täytyy lisäksi määrittää operointiasetus ja looginen taso, joista operointiasetuksella määritetään onko pinni lähettävässä vai vastaanottavassa tilassa sekä looginentaso kertoo onko se käynnistyessä jännitteellinen. Merkkivalojen toiminnan kannalta halutaan että LEDien looginen taso on nolla prototyypin käynnistyessä, sekä ne on asetettu lähettävään tilaan. Yksinkertaisin tapa muuttaa haluttujen pinnien asetuksia on käyttää Raspberrylle tehtyä gpio -ohjelmaa. Esimerkiksi GPIO18 asettaminen lähettäväksi onnistuu ’gpio -g mode 18 out’ -käskyllä sekä ’gpio -g write 18 0’ muuttaa pinnin jännitteettömäksi. Nämä käskyt kun lisätään /etc/rc.local -tiedostoon muiden pinnien määrittelyiden ohessa, voidaan varmistaa että merkkivaloja käyttämät pinnit ovat valmiina ohjattavaksi alustan käynnistyessä. [33] [34] [35] [36]

6.3 Huawein VRP komennot

Huawei VRP käskyrakenne on monoliittinen, eli laitemäärittelyt koostuvat sanatarkoista komennoista käyttöliittymässä. Virheellisesti muodostetun käskyn antaminen antaa virheilmoituksen eikä käskyä suoriteta. Huawein konfiguroimiseksi on kaksi perustilaa, User View ja System View. User View on perustila, jossa käyttäjä voi vain tarkastella laitteen toiminnallisia asetuksia ilman että käyttäjä pääsisi muuttamaan sen toiminnallisia asetuksia. Verkkolaitteen konfiguroimiseksi tarvitaan kuitenkin käyttäjältä laajennetut oikeudet eli pääsyn System View -tilaan. Oletuksena System View -tilaan pääsee vain admin -käyttäjä, mutta siihen päästäkseen on annettava ’system-view’ -komento, jonka onnistumisen jälkeen komentorivin teksti rajataan hakasulkeilla normaalien operaattorimerkkien sijasta. Konsoliesimerkissä 6.1 nähdään miten käyttäjä havaitsee tilanmuutoksen käyttöliittymässään. Huawei VRP tukee myös ASCII -merkistön erikoismerkkejä, joita voidaan syöttää yhdistämällä CTRL -painike ja kirjain. System View tilasta siirtymistä takaisin perustilaan pääsee ’quit’ -käskyllä tai hyödyntämällä ASCII erikoismerkkiä Ctrl+Z kuten konsoliesimerkin 6.1 rivillä 8 on käytetty.

Esimerkkinä User View -tilassa voidaan katsoa porttien tilat komennolla ’display interface brief’. Tämä komento antaa kaikki normaalin toiminnan kannalta oleellisimmat tiedot kytkimen porttien tiloista. Yksittäisen portin asetuksia pääsee muokkaamaan ensin siirty-

Ohjelma 6.1. *Siirtyminen User View -tilasta System view tilaan Huawei VRP käyttöjärjestelmässä.*

```
1 <Quidway>
2 <Quidway>system-view
3 Enter system view, return user view with Ctrl+Z.
4 [Quidway]quit
5 <Quidway>
6 <Quidway>system-view
7 Enter system view, return user view with Ctrl+Z.
8 [Quidway] ## Syotetaan nappainyhdistelma  ctrl+Z
9 <Quidway>
```

Ohjelma 6.2. *Verkkolaitteen minimi verkkoasetusten määrittäminen.*

```
1 <Quidway>system-view
2 [Quidway]interface Vlanif1
3 [Quidway-Vlanif1]ip address 1.1.1.1 255.0.0.0
4 [QuidwayVlanif1]quit
5 [Quidway]ip route-static 0.0.0.0 0.0.0.0 1.1.1.10
6 [Quidway]quit
7 <Quidway>
```

mällä System View -tilaan ja sitten lähettämällä esimerkiksi käsky 'interface GigabitEthernet 0/0/1', jonka myötä voidaan kyseisen portin asetuksia muokata. Annetut muutokset konsolissa vaikuttavat välittömästi kytkimen toimintaan rautatasolla eikä tehtyjä muutoksia tarvitse erikseen suorittaa. Verkkoasetusten määrittämisessä täytyy laitteelle syöttää useita yksilöiviä asetuksia kuten esimerkiksi IP- ja aliverkko-osoite sekä laitenimi. Esimerkissä 6.2 määritellään Huawei VRP verkkolaitteelle minimi verkkoasetukset IP -osoitteella ja verkkomaskilla, jonka jälkeen laitteelle teoriassa voi yhdistää muualta samasta verkkoalueesta.

6.4 Merkkien ja komentorivien lähettäminen ja vastaanottaminen Pythonissa

Prototyypin ohjelma päätettiin alkuvaiheessa toteuttamaan Pythoniin tehdyllä PySerial -moduulilla. PySerial -moduulilla voidaan lukea ja kirjoittaa sarjaportin datalinjoja sekä asynkronisen tiedonsiirron vuonhallinta on myös tuettu. Kommunikointi vastapään verkkolaitteen konsoliporttiin tapahtuu esimerkiksi write() ja read() -käskyillä. Write() -käskyä voidaan käyttää pääasiassa kolmella tavalla, joka tekee siitä monipuolisen tavan lähettää tietoa vastapäälle. Oletuksena käskyllä lähetetään merkki kerrallaan binäärimuodossa, mutta lisäämällä alkuun '\x' -merkintä, voidaan myös käyttää ASCII -merkistön hexadesimaalikoodeja. Esimerkiksi komennolla write('\x73') lähettää PySerial sarjaportista pa-

Ohjelma 6.3. *Yksinkertaistettu Huawei VRP -käyttöjärjestelmän verkkoasetusten määrittäminen Pythonin ja PySerialin avulla.*

```
1 def code():
2     ser.write(b'system-view\n')
3     write(b'int vlan 1\n')
4     write(b'ip address 1.1.1.1 255.0.0.0\n')
5     write(b'ip route-static 0.0.0.0 0.0.0.0 1.0.0.10\n')
6     write('\x1A') # Lahettaa CTRL+Z
```

ketin, joka sisältää ASCII taulukon paikassa 0x73 olevan s -merkin. Kolmas ja ohjelman kannalta käytännöllisin tapa on lisätä lisämääritelmä 'b', joka mahdollistaa kokonaisten tekstirivien lähettämisen yhdellä komennolla. System View -komento voidaan täten lähettää yhdellä write(b'system view\n') -komennolla. Komennon loppuun lisätään rivinvaihtomerkki '\n', jonka vastaanottava pää tulkitsee komennon suoritusmerkkinä ja niin sitä ei tarvitse erikseen lähettää. Yksittäisten merkkien lähettäminen tulee aiheelliseksi kun halutaan käyttää ASCII -merkistössä löytyviä ohjausmerkkejä. Ohjelmallisesti on hankalaa arvioida missä tilassa verkkolaitteen konsoli on tai onko edes yhteys laitteeseen enää aktiivinen. ASCII -ohjausmerkkien lähettämisellä voidaan esimerkiksi palauttaa verkkolaitteen komentorivi takaisin juuritasolle uusien käskyjen suorituksen onnistumiseksi tai pelkästään yhteyden toimiminen voidaan varmistaa lähettämällä edellä mainittu '\n' -rivinvaihtomerkki verkkolaitteelle. Sarjaporttiin tulevan datan lukeminen voidaan toteuttaa kahdella eri tavalla, joko lukemalla tietty määrä puskurissa olevia merkkejä tai tiettyyn merkkiin asti. Read_until() lukee merkkejä kunnes se vastaanottaa tietyn merkin, joka oletuksena on '\n' -rivinvaihtomerkki ja tämä tapa soveltuu hyvin konsolista vastaanotettavien logitulosteiden lukemiseksi. Read() taas lukee sulkujen sisään määritellyn määrän merkkejä, jolloin se soveltuu tilanteeseen milloin rivinvaihtoa ei tule. Kummatkin tottelee kuitenkin koodissa 6.4 määriteltyä 'timeout' -arvoa, jonka jälkeen koodin suoritus jatkuu. Prototyypin ohjelmakoodi toteutettiin pääasiassa read() -käskyllä, joka tyhjentää tietyn määrän merkkejä puskurista ja arvioi lukemastaan missä tilassa verkkolaitteen konsoli sillä hetkellä on. Teoriassa verkkolaitteen ohjelmoimiseksi ei tarvitse kuin lähettää suoritettavat komentorivit vastapäälle, mutta koska komentojen suorittaminen vaatii että verkkolaite on tietyssä tilassa on tiedettävä milloin komennot lähetetään. Ohjelmassa 6.3 on lyhyt esimerkki verkko-osoite määritysten syöttämisestä Huawei VRP käyttöjärjestelmään PySerialin avulla. Verkkoasetusten määrittäminen tarvitsee laajennetut oikeudet eli aluksi ohjelmalla lähetetään system-view käsky. Tämän jälkeen asetukset voidaan lähettää yksi rivi kerrallaan ja lopuksi CTRL+Z käytetään palaamaan perustilaan.

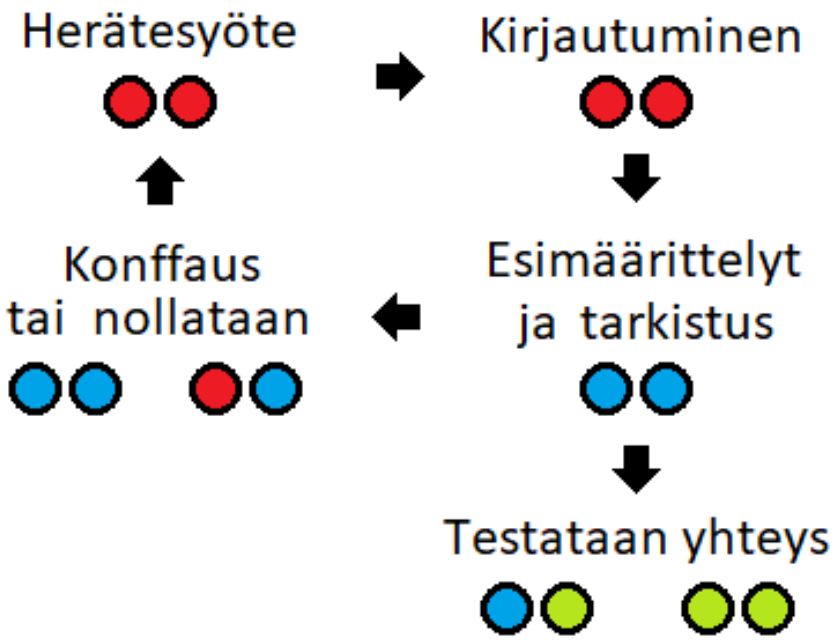
Esimerkki sarjaportista luetun syötteen käsittelylle löytyy koodista 6.5 ja saadun syötteen mukaan päätellään missä tilassa verkkolaite sillä hetkellä on. [37]

6.5 Merkkivalot ja niiden logiikka

Prototyyppiin luodun ohjelman eri tilat ilmaistaan käyttäjälle kahden RGB -ledin avulla eli yhdessä on punaista, sinistä ja vihreää valoa emittoivat elementit. Valojen avulla prototyyppi ilmaisee käyttäjälleen missä vaiheessa ohjelman suoritus on milloinkin, jolloin hän pystyy karkeasti havaitsemaan mitä esiohjelmoija tekee. Yksittäisellä valolla tai niitä yhdistämällä voitaisiin muodostaa kymmeniä eri värisävyjä, mutta käytön helpottamiseksi ei käytetä kuin LEDien omia värejä. LEDien värijärjestys halutaan myös pitää yksiselitteisenä ilman ettei käyttäjän tarvitse miettiä kumpi valoista on vasen ja kumpi oikea.

Kuvassa 6.2 esitetään ohjelman eri päävaiheiden lisäksi värijärjestykset. Punaisella värillä ilmaistaan että yhteys tai verkkolaite on sellaisessa tilassa missä sen ei pitäisi olla. Sininen antaa käyttäjälle tiedon että ohjelman suoritus on onnistunut, mutta osa siitä on vielä kesken. Vihreällä voidaan viestiä verkkolaitteen ohjelmoinnin onnistumisesta. Ohjelman toimiessa odotetusti on molemmat värit samaa sävyä. Kahden eri värin näyttämisellä voidaan tarkemmin kertoa mitä ohjelma tekee; punaisen ja sinisen esittämisellä viestitään laitteen olevan vajaasti määritelty ja ohjelma pyrkii nollaamaan määrittelyitä. Toinen missä tätä hyödynnetään on yhteyden testaaminen, jonka ansiosta käyttäjä tietää onko verkkolaitteella yhteys muuhun osaan verkkoa.

Ohjelmiston kehittyessä on Raspberry Pissä mahdollista käyttää erillistä näyttöä esittämään tarkempia virhekoodeja ja mahdollisesti konfiguroidut verkkoasetukset.



Kuva 6.2. *Prototyypin ohjelman toiminnalliset vaiheet sekä miten merkkivalot muuttuvat niiden mukaan*

Ohjelma 6.4. *Prototyypin ohjelmakoodissa tarvittavat vakiot ja kirjastot sekä konsoliportin asetusten alustus.*

```
1 import pexpect, sys, time, serial
2 CONSOLE_PORT = '/dev/serial0'
3 SERIAL_SPEED = 9600
4 PRE_LOGIN_LINE = ['Username:','Password:']
5 QUERY = ['[Y/N]:','[Y/N]: ']
6 ALREADY_LOGGED = ['>','] ']
7 ser = serial.Serial(CONSOLE_PORT, SERIAL_SPEED, timeout=1)
```

6.6 Ohjelmakoodin toteutus

Prototyypin ohjelmakoodin tehtävänä on pystyä ohjelmoimaan tehdasasetuksissa oleva Huawei S -sarjan kiinteistökytkin ennalta määriteltyihin verkkoasetuksiin sekä tyhjentämään vajaasti määritelty laite takaisin tehdasasetuksille. Mahdollisen käytössä olevan laitteen tunnistaminen on myös tärkeää ettei käyttäjä onnistu vahingossa tyhjentämään tärkeän verkkolaitteen asetuksia, mutta tämä on sinänsä tarpeetonta sillä verkkolaitteen salasana on jo vaihdettu käyttöönotetussa laitteessa. Kuvassa 6.2 esitetään toteutetun ohjelman vaiheet, jotka käydään jokaisen verkkolaitteen määrittelyssä läpi. Käytännössä ohjelman suoritus ei ole näin suoraviivainen, vaan ohjelman seuraava vaihe määräytyy sen mukaan mitä tietoa on verkkolaitteesta saatu. Tämä käytännössä tarkoittaa että ohjelman perustilaan eli odotustilaan palataan jokaisen tehtävän päätteeksi. Onnistuneen ohjelmoinnin päätteeksi verkkolaite käynnistetään alusta ja konfiguraatio tarkistetaan uudestaan. Konfiguraation ollessa kunnossa siirrytään yhteystilaan, jossa laite yrittää tavoittaa oman osoitteensa yhdyskäytävää.

6.6.1 Ohjelman alustaminen ja vakiot

PySerialin asetuksiin asetettiin 9600 bitin sekuntinopeus, joka on Huawein S5300 -sarjan verkkolaitteen konsolin oletusnopeus ja lisäksi konsoliportiksi määriteltiin 'serial0', jonka mahdollistaa sarjaportiksi määritellyn UART -liitännän käyttämisen.

Ohjelmakoodissa 6.4 asetetaan myös ohjelmalle tarvittavat kirjastot ja luodaan koodia selkeyttäviä vakioita. PRE_LOGIN_LINE kirjautumisvakioissa on tallennettu Huawei VRPn kirjautumissyötteet. Komentosyötteet on tallennettu ALREADY_LOGGED vakioon ja ne kertovat miten kirjautunut käyttäjä näkee komentorivin päättyvän. Näiden vakioiden avulla pystytään päättelemään missä tilassa verkkolaitteen konsoli milloinkin on. Lopuksi luodaan 'ser' -muuttuja, jota käytetään tekstin lukemiseen ja lähettämiseen sarjaportin avulla.

Ohjelma 6.5. *Odotustila eli ohjelman perustila ja aliohjelmiin siirtyminen.*

```
1  line = ser.read(1000)
2  if len(line) > 0:
3      if any(line.endswith(end) for end in PRE\_LOGIN\_LINE):
4          if login(ser, line):
5              mode\_selection(ser)
6      elif line.endswith(tuple(ALREADY\_LOGGED)):
7          script\_mode(ser)
8          mode\_selection(ser)
9      else:
10         if line.endswith('Continue␣to␣set␣it?␣[Y/N]:␣'):
11             ser.write(b'y\n')
12             time.sleep(1)
13             ser.write(b'salasana\n')
14             ser.write(b'salasana\n')
15             ser.write(b'quit\n')
16             continue
17         elif line.startswith('\r\nError:'):
18             ser.write(b'n\n')
19             continue
20         ser.write(b'\n')
21         time.sleep(3)
22 else:
23     ser.write(b'\n')
24     time.sleep(3)
```

6.6.2 Odotustila

Odotustila toimii pääasiallisena perustilana mihin ohjelma palaa jokaisen toiminnan jälkeen. Aluksi sarjaportin puskurista luetaan enintään 1000 merkkiä muuttujaan. Muuttujan viimeisiä sanoja tai kirjaimia verrataan tiettyihin vakioihin, joiden täsmätessä suoritetaan tietyt operaatiot. Syy miksi puskurista luetaan paljon enemmän merkkejä on puskurin tyhjentäminen turhista käynnistyksenaikaisista tulosteista ja muista konsolin kautta tulleista merkeistä. Muuttujan päättyessä oikeisiin tulosteisiin suoritetaan tietty toimenpide, joka voi olla kirjautumiskysely tai komentojen syöttörivi. Tilanteessa jossa puskuri on tyhjä lähettää ohjelma rivinvaihdon testatakseen yhteyden toimivuuden ja herättääkseen verkkolaitteen konsolin. Yhteys on todennäköisesti poikki jos laitteelta ei saada syötettä. Ideaalitapauksessa rivinvaihdon lähettäminen saa vastaukseksi rivin, joka on kirjautumiskysely tai kirjautumisen jälkeinen komentorivi.

Ohjelmakoodissa 6.5 näkyy kuinka odotustila vertailee muuttujan syötettä. Luetun muuttujan päättyessä kirjautumisvakioihin, voidaan siirtyä LOGIN -aliohjelmaan tai jos muuttuja loppuu komentovakioihin, oletetaan laitteen olevan valmis vastaanottamaan käskyjä. Poikkeuksena näiden kahden vaihtoehdon lisäksi tulee nollatun laitteen salasanan vaih-

Ohjelma 6.6. *Konsolikirjautuminen Huawei VRP -järjestelmään.*

```
1 while(1):
2     if line.endswith(PRE_LOGIN_LINE[0]):
3         ser.write(b'user\n')
4         line = ser.read(100)
5     if line.endswith(PRE_LOGIN_LINE[1]):
6         ser.write(b'salasana\n')
7         line = ser.read(100)
8         script_mode(ser)
9         return 1
```

Ohjelma 6.7. *Huawei VRP -käyttöjärjestelmään tehtävät esimäärittelyt kommunikoinnin helpottamiseksi.*

```
1 def script_mode(ser):
2     ser.write(b'system-view\n')
3     ser.write(b'info-center␣disable\n')
4     ser.write(b'user-interface␣console␣0\n')
5     ser.write(b'screen-length␣0\n')
6     ser.write('\x1A')
7     time.sleep(1)
```

dos, jonka myötä asetetaan laitteelle uusi salasana. Käytännössä salasananvaihdon ollessa kesken aiheuttaa rivivaihdon syöttäminen virheviestin, josta pääsee pois vastaamalla kieltävästi 'n'.

6.6.3 Kirjautuminen ja esimäärittelyt

Kirjautuminen toteutetaan omassa aliohjelmassaan sen mukaan missä vaiheessa se on. Ohjelmakoodissa 6.6 ensin tarkistetaan haluaako verkkolaite kirjautumistiedot vai pelkän salasanan.

Onnistuneen kirjautumisen jälkeen asetetaan skriptausta helpottavat asetukset laitteelle. Ohjelmakoodissa 6.7 esitettyjen skriptausasetusten asettaminen on tärkeää konsoliportin syötteiden lukemiselle, koska oletuksena Huawei VRP tulostaa logi tulosteita konsoliporttinsa kautta ja nämä satunnaisesti tulevat ilmoitukset voivat sekoittaa ohjelman toimivuuden. Lisäksi päältä otetaan pois oletusasetus, joka rajoittaa kerralla tulostettavien rivien määrän kahteenkymmeneenviiteen. Tämän myötä laitteelle syötettävien komentorivien tulokset tulostuvat kokonaisuudessaan read() -funktiolle.

Ohjelma 6.8. *Esimerkki verkkoasetusten lähettämisestä verkkolaitteelle.*

```
1  ser.write(b'system-view\n')
2  ser.write(b'sysname laitenimi\n')
3  ser.write(b'port hybrid tagged vlan 1\n')
4  ser.write(b'interface giga0/0/1\n')
5  ser.write(b'vlan 1\n')
6  ser.write(b'quit\n')
7  ser.write(b'int vlan 1\n')
8  ser.write(b'ip address 1.1.1.1 255.255.255.0\n')
9  ser.write(b'ip route-static 0.0.0.0 0.0.0.0 1.1.1.10\n')
10 ser.write(b'rsa local-key-pair create\n')
11 time.sleep(2)
12 ser.write(b'512\n')
13 ser.write(b'aaa\n')
14 ser.write(b'local-user admin password cipher salasana\n')
15 ser.write(b'local-user admin ftp-directory flash:\n')
16 ser.write('\x1A')
17 time.sleep(20)       # Konfigurointi voi olla kesken
18 ser.flush()
19 ser.write(b'save laite.cfg\n')
20 ser.write(b'y\n')
21 ser.write(b'startup saved-configuration laite.cfg\n')
```

6.6.4 Verkkoasetusten määrittäminen

Ohjelmassa 6.8 on esitetty yksinkertaistettu tapa määrittää verkkoasetukset PySerialin avulla. Laiteyksilöinti on pääosin suoraviivainen toimenpide, mutta niiden onnistunut lähettäminen vaatii ettei laitteeseen ole ennalta määritelty muita asetuksia. Lisäksi verkkolaitteen ohjelmistoversio usein vaikuttaa missä komentoformaatissa laite hyväksyy määrittelyitä, mikä voi hankaloittaa tämän käyttöönottoa. Verkkolaitteelle määritellään esimerkiksi nimi, IP -osoite, sallitut protokollat, uudet kirjautumisasetukset sekä paikallisen RSA -avainparin pituus. Salasana tässä tapauksessa pidetään muuttumattomana, mutta jos sen haluaa muuttaa on se otettava huomioon seuraavassa kirjautumisessa. Lopuksi laitemäärittely rivien lähettämisen jälkeen on konfiguraatio tallennettava ja asetettava käynnistyksen aikaiseksi konfiguraatiotiedostoksi.

6.6.5 Komentojen lähettäminen ja tulosteiden käsittely

Verkkolaitteelle lähetetyt komennot hoidetaan kootusti yhden funktion avulla. Tämä on toteutettu 6.9 ohjelmassa esitellyn read_output() -funktiolla, jonka pääasiallisena tehtävänä on tallentaa etsittävän tiedon sisältämät rivit Osumat -taulukkoon. Toisena tehtävällä funktiolla on etsiä luetulta riviltä virheellisiä tulosteita, joiden myötä komennon suorittaminen ei ole onnistunut halutulla tavalla. Virheellisen rivin tapauksessa se tallennetaan taulu-

Ohjelma 6.9. *Komennon suorittamiseen ja tiedon tallentamiseen tarkoitettu aliohjelma.*

```
1  def read_output(ser, trigger, command, osumat):
2      exec('ser.write(b\x27'+str(command)+'\\n\x27)')
3      kasky = ser.readline()
4      while True:
5          line = ser.readline()
6          if trigger in line:
7              if trigger == 'Info: System is rebooting...'
8                  osumat.append('BOOT')
9                  break
10             osumat.append(line)
11         elif line.endswith(tuple(ALREADY_LOGGED)):
12             break
13         elif line.endswith('[Y/N]:') or line.endswith('[Y/N]'):
14             osumat.append(line)
15             break
16         elif line.startswith('Error:'):
17             osumat.append(line)
18             break
19         elif line is None and osumat == []:
20             osumat.append('0')
```

kon loppuun ja palautetaan muiden osumien mukana. Funktiolle annetaan 'command' -muuttuja, joka sisältää suoritettavan komennon sekä 'trigger' -muuttujan, joka sisältää tekstin mitä komennon aiheuttamasta tulosteesta etsitään. Lopuksi Osumat -taulukko palautetaan käsittelemättä sitä suorittavalle ohjelmalle, minkä tehtäväksi jääkin varsinainen tiedonkäsitteleminen.

Write() -funktion jouduttiin suorittamaan Exec() -funktion avulla, koska silloin tämän sisältämät erikoismerkit saatiin tekstimuodossa tulostettua oikein. Tämän jälkeen readline() -funktiolla poistettiin äsken suoritettu komentorivi ettei sen mahdollisesti sisältämät osumat vaikuttaisi rivien palauttamiseen. Read_output() -funktiota käytetään verkkolaitteen konfiguraatioasetusten hakemisessa, konfiguraatiotiedoston nollauksessa sekä laitteen uudelleenkäynnistyksessä.

6.6.6 Verkkolaitteen testaus

Ohjelma 6.10 käyttää read_output() -funktiota suorittamaan verkkolaitteella tiettyjä komentoja, joiden avulla testataan konfiguraatiotiedostojen määrittelyt käynnistykseen sekä tarpeellisten verkkoasetusten löytyminen. Määrittelyjen tarkastamisella halutaan varmistaa että laite on konfiguroitu ja että sen asetuksista löytyvät kaikki tarvittavat asetukset.

Aluksi tarkistetaan löytyykö laitteelta .cfg vai .zip -päätteinen tiedosto, joista .zip löytyminen merkitsee verkkolaitteen käynnistyneen tehdasasetuksilla. Konfiguraatiotiedoston

Ohjelma 6.10. *Verkkolaitteen konfiguraation tarkistava ohjelmakoodi.*

```
read_output(ser, '<Quidway>\r\n', '', osumat)
if osumat != []: return 0   # Konsoli ei perustilassa
read_output(ser, '.cfg', 'dir', osumat)
if len(str(osumat)) is not 0 :      # jos cfg loytyy
    #   Tarkistetaan onko cfg seuraavassa kaynnistyksessa
    read_output(ser,
    'Next startup saved-conf','display startup', seuraava)
    if seuraava in osumat:
        #   cfg asetettu seuraavaan kaynnistykseen
        #   Tarkistetaan kyseinen cfg
        # Onko IP maaritetty
        read_output(ser, 'ip address 1.',
            \'display current | include ip' , out)
            ip = word
            else:   return -1
        # Onko VLAN maaritetty
        read_output(ser, 'interface Vlanif1',
            \'display current | include Vlanif1' , out)
            vlanif1 = 1
            else:   return -1
        # Onko VLAN1 maaritetty portille
        read_output(ser, port hybrid tagged vlan 1',
            \'display current | include tagged vlan 1' , out)
            tagged = 2
            else:   return -1
        # Jos maarittelyt loytyvat, on tarkistus kunnossa
        if ip.startswith('1.1.1.') and vlanif1 and tagged == 2:
            return 1    # Laite maaritelty oikein
            else:   return -1
    else:   return 2    # Laite vasta nollattu -> vaatii bootin
else:
    #   Laite tehdasasetuksissa
```

löytyessä voidaan olettaa että verkkolaitetta on jo konfiguroitu, jolloin verkkoasetusten testaaminen voidaan aloittaa. Jos kumpaakaan tiedostoa ei löydy, on laite todennäköisesti jo nollattu ilman uudelleenkäynnistystä. Uudelleenkäynnistys vaaditaan ennen kuin laitteen määrittelyitä voidaan tehdä. Ohjelman 6.10 koodista on poistettu käsiteltävän taulukon rivinkäsittelyt, joilla parsitaan tulosteista haluttu informaatio muuttujaan. Ohjelman palauttamasta numerosta päätellään tarvitseeko verkkolaite käynnistää uudelleen, nollata sen käynnistystiedost vai onko sen esikonfiguraatio valmis.

Ohjelma 6.11. *Huawei VRP asetustiedoston nollaaminen ohjelmalla.*

```
1  def reset_device(ser, pre_conf):
2   if pre_conf is 1:
3    # Laite on maaritelty oikein, halutaanko nollata?
4   else:
5    read_output(ser,'Warning:␣...␣delete␣saved␣conf',\
6    'reset␣saved-configuration',osuma)
7    if osuma[-1].endswith('[Y/N]:'):
8     read_output(ser,'Warning:␣Now␣clearing␣conf','y',osuma)
9     if osuma[-1].startswith('Info:␣Succeeded␣clearing␣conf')\
10    or osuma[-1].startswith('Error:␣config␣not␣exist.'):
11     reboot(ser)
12    else:
13     reboot(ser) # Siirrytaan uudelleenkaynnistysohjelmaan
```

6.6.7 Nollaustila

Ohjelmassa 6.11 esitellään nollaustila, jossa poistetaan vanha konfiguraatiotiedosto ja siirrytään uudelleenkäynnistysohjelmaan. Huawein käyttöjärjestelmä kysyy käyttäjältä varmistuksen tiedoston poistamisesta, mutta määrittelytiedoston puuttuessa se ei anna mitään varmistusta. Määrittelyn toteutumiseksi on lopuksi laite käynnistettävä uudelleen ja se toteutetaan erillisessä reboot() -ohjelmassa.

6.6.8 Uudelleenkäynnistys

Huawein ensimmäistä kertaa käynnistyksessä laite luo vrpcfg.zip tiedoston, johon se tallentaa verkkolaitteen perusmäärittelyt. Laitenollauksen jälkeen on konfiguraatio tiedostoksi tallennettu 'none', jonka takia se ennen uudelleenkäynnistystä tarkistaa löytyykö kyseistä zip -tiedostoa. Mahdollisesti kyseistä tiedostoa voidaan käyttää konfiguraation tallennukseen, mutta tässä ohjelmassa käytetään laite.cfg -tiedostoa määrittelyiden tallentamiseen. Uudelleenkäynnistys toteutetaan ohjelmassa 6.12, joka huomioi käyttöjärjestelmän mahdolliset kysymykset ennen toimenpiteen toteuttamista.

Määrittelyiden tallentamisen jälkeen tehtävän uudelleenkäynnistyksen yhteydessä Huawei kysyy käyttäjältä halutaanko suorituksessa oleva konfiguraatio tallentaa, johon varmuuden vuoksi vastataan kieltävästi.

6.6.9 Yhteystila

Yhteystilassa ohjelma tarkistaa verkkolaitteelta onko tietty portti aktiivinen. Portin ollessa aktiivinen voidaan olettaa laitteen olevan yhteydessä muuhun verkkoon, jonka jälkeen ohjelma ns. 'pingaa' määrittelyissä asetettua 'ip route-static' -osoitetta. Kyseinen toimen-

Ohjelma 6.12. *Huawei VRP järjestelmän uudelleenkäynnistävä ohjelmakoodi.*

```
1  def reboot(ser):
2      read_output(ser,\
3      'Info: The system is now comparing conf','reboot',osuma)
4      if osuma[-1].startswith('Warning: The conf has modified'):
5          read_output(ser,\
6          'Info: If want to reboot with diagnostic' ,\
7          'n' ,osuma)
8          if osuma[-1].startswith('vrpcfg.zip exists, overwrite?'):
9              read_output(ser,\
10             'Info: If want to reboot with diagnostic',\
11             'n',osuma)
12         if osuma[-1].startswith('System reboot! Continue?'):
13             read_output(ser,\
14             'Info: System is rebooting, please wait...' ,\
15             'y',osuma)
16             if osuma[0] == 'BOOT':
17                 # Laite kaynnistyy uudelleen
18                 # siirrytaan takaisin Odotustilaan
```

Ohjelma 6.13. *Yhteyden toiminnan testaus.*

```
1  <Huawei> ping 1.1.1.10
2    PING 1.1.1.10: 56  data bytes, press CTRL_C to break
3      Reply from 1.1.1.10: bytes=56 Sequence=1 ttl=64 time=1 ms
4      ...
5      Reply from 1.1.1.10: bytes=56 Sequence=1 ttl=64 time=1 ms
6
7    --- 1.1.1.10 ping statistics ---
8      5 packet(s) received
```

pide on esitetty ohjelmassa 6.13, jossa Huawei lähettää 3.3.3.3 -osoitteeseen 5 kappaletta ICMP -paketteja ja yhteyden ollessa kunnossa, vastapää vastaa jokaiseen pakettiin omalla paketillaan.

Toimimattomassa yhteydessä tulostuu 'reply from..' -rivin tilalla 'Request time out' -teksti. Asennuksen kannalta tämä toimenpidettä ei pidetä tärkeänä ja sitä ei nykyisessä prototyypissä ole toteutettuna. Verkkolaitetta asennettaessa voi olla hyvin todennäköistä ettei asennuskohteessa ole vielä toimivia yhteyksiä ja tämän takia riittää että ohjelma pystyy varmistamaan konfiguraation olevan kunnossa.

7 YHTEENVETO

Verkkolaitteiden asetusten määrittämisessä suurin aika menee sen mahdollistamiseen. Kun laite on saatu käynnistettyä on myös käynnistettävä konsoliyhteyden mahdollistava tietokone USB-RS232 adaptereineen, jotta verkkolaitteen käyttöjärjestelmään edes päästään määrittelemään mitään. Tämän lisäksi on tiedettävä miten kirjautua laitteelle sekä mitä asetuksia sinne täytyy asettaa. Pelkkien esimäärittelyiden tekemiseen voikin tuhlautua huomattava osa asennukseen käytetystä ajasta, vaikka kaikki yhteyden mahdollistamiseen tarvittavat oheislaitteet löytyisivätkin ja toimisivat valmiiksi. Kustannuksissa säästäminen on nykyaikaa ja varsinkin sellaisille tahoille, joiden liiketoiminnan tärkeä osa tuoda uusia laitteita verkkoon, kuten operaattoreilla on järkevää suoraviivaistaa toimintamalliaan kustannusten vähentämiseksi. Lisäkustannuksia ilmaantuu jos verkkolaite osoittautuukin olevan tehdasasetuksissaan tai väärin konfiguroiduksi, jolloin asentaja joutuu tarkemmin perehtymään laitteen käyttöjärjestelmään. Ammattitaitoa väheksymättä on valitettavan yleistä vajavaiset tietotekniset taidot mahdollisten ongelmien sattuessa.

Tässä työssä luodun prototyypin tarkoituksena onkin ensinnäkin helpottaa asentajan työskentelyä ja samalla nopeuttaa verkkolaitteiden asennusta. Esiohjelmoijan hyödyt teoriatasolla ovat ilmiselvät; asentajalle annetaan asennettavan verkkolaitteen mukana työkalu, joka voidaan jo ennen laitteen sähköistämistä kytkeä sen konsoliporttiin ilman minkäänlaista ymmärrystä verkkolaitteen konfiguroimisesta. Kuitenkin esiohjelmoijan mukana toimitetaan käyttöohjeiden lisäksi tarpeelliset kaapelit eri verkkolaitteiden määrittelemiseksi. Esiohjelmoijan rakentamisesta aiheutuneet kustannukset on nopeasti kuittaantuneet, jos sitä käytetään edes muutamassa asennuksessa onnistuneesti. Prototyypin ohjelmaa kehittäessä suurin haaste oli luoda sarjaportin kommunikaation mahdollistavat aliohjelmat, joiden jälkeen niitä hyödynnettiin määrittelyitä tekevissä aliohjelmissa. Todennäköisesti ohjelmiston tuen laajentaminen muille laitemalleille vaatii laitekohtaisten aihioiden luomisen, joiden avulla pystytään helposti lisäämään ohjelmistoon uusia verkkolaitteita. Ohjelman toimimattomuus yleensä johtui Huawei VRP -käyttöjärjestelmän tapauksessa yksittäisistä kysymyksistä ja kehotteista mitä järjestelmä käyttäjälle lähetti ja niiden korjaamisessa riitti sen huomioon ottaminen rivejä lukiessa.

Prototyyppiä on tähän mennessä testattu muutaman Huawei S -sarjan kytkimen asennuksessa hyvällä menestyksellä ilman että sen ohjelmaan olisi tarvinnut tehdä muutoksia. Menestyäkseen koko esiohjelmoija projekti vaatii jatkuvaa ylläpitoa uusien ohjelmis-

toversioiden takia. Valitettavan yleistä on että valmistajat lähettävät laitteitaan vaihtelevilla ohjelmistoversioilla, joiden komennoissa ja käyttäytymisessä ilmenee jonkin verran eroja.

LÄHTEET

[1] Tang, S. *Advantages of Ethernet vs. SONET/SDH*. TC Communications Inc. Irvine, CA, 2015. URL: https://www.tccomm.com/Content/pdf/Literature/White-Paper-Ethernet-Advantages.pdf.

[2] *Access Technologies for Carrier Ethernet*. 6. joulukuuta 2018. URL: https://wiki.mef.net/display/CESG/Access+Technologies+for+Carrier+Ethernet (viitattu 24. 06. 2020).

[3] *Why MEF*. 24. kesäkuuta 2020. URL: www.mef.net/about-mef (viitattu 24. 06. 2020).

[4] *BroadbandEdge NetworkDesign*. JuniperNetworks, Inc. Sunnyvale, CA, 2016. URL: https://www.juniper.net/documentation/en_US/release-independent/solutions/information-products/topic-collections/broadband-edge-ref-arch/broadband-edge-ref-arch.pdf.

[5] *USB Data Packet Structure*. Future Technology Devices International Limited. Glasgow, 2009. URL: https://www.ftdichip.com/Support/Documents/TechnicalNotes/TN_116_USB%5C%20Data%5C%20Structure.pdf.

[6] *IBM Parallel Port FAQ/Tutorial*. 9. tammikuuta 1994. URL: http://massmind.org/techref/io/parallel/faq0994.htm (viitattu 24. 06. 2020).

[7] *UART | Serial Communication With PIC Microcontrollers Tutorial*. 11. syyskuuta 2019. URL: https://deepbluembedded.com/uart-pic-microcontroller-tutorial/ (viitattu 24. 06. 2020).

[8] Fe, A. *Digitaalitekniikka (piirit), Rekisterit ja laskurit*. Metropolia. 2014. URL: http://users.metropolia.fi/~koiva/Rob_dig/Materiaali/18Rekisterit_ja_laskurit.pdf.

[9] *Networking and Communications*. 24. kesäkuuta 2020. URL: http://archive.fabacademy.org/archives/2017/fablabtoscana/students/207/exercise15.html (viitattu 24. 06. 2020).

[10] *Difference Between DTE and DC*. 4. syyskuuta 2017. URL: https://techdifferences.com/difference-between-dte-and-dce.html (viitattu 24. 06. 2020).

[11] *BCM2835 ARM Peripherals*. Broadcom. Science Park Milton Road Cambridge, 2012. URL: https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf.

[12] *UART configuration*. 11. syyskuuta 2019. URL: https://github.com/raspberrypi/documentation/blob/master/configuration/uart.md (viitattu 24. 06. 2020).

[13] *PrimeCell® UART (PL011) Technical Reference Manual*. Texas Instruments Inc. 2010. URL: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0183g/index.html.

[14] *Interface Circuits for TIA/EIA-232-F*. Texas Instruments Inc. 2002. URL: https://www.ti.com/lit/an/slla037a/slla037a.pdf.

[15] *AN-917 Popular Connector Pin Assignments for Data Communication*. Texas Instruments Inc. 2004. URL: https://www.ti.com/lit/an/snla039/snla039.pdf.

[16] *RS232 serial null modem cable wiring*. 1. marraskuuta 2019. URL: https://www.lammertbies.nl/comm/info/rs-232-null-modem (viitattu 24. 06. 2020).

[17] *MAX32323-V to 5.5-V MultichannelRS-232Line Driver/ReceiverWith 15-kV ESD Protection*. Texas Instruments Inc. 2017. URL: https://www.ti.com/lit/ds/symlink/max3232.pdf.

[18] *Tech Stuff - RS-232 Cables, Wiring and Pinouts*. 6. huhtikuuta 2016. URL: https://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm (viitattu 24. 06. 2020).

[19] *Yost Serial Device Wiring Standard*. 2. helmikuuta 2020. URL: http://yost.com/computers/RJ45-serial/ (viitattu 24. 06. 2020).

[20] *ASCII Code Table*. 23. elokuuta 2015. URL: https://www.ieee.li/computer/ascii.htm (viitattu 24. 06. 2020).

[21] *ASCII Code Chart*. 21. heinäkuuta 2012. URL: https://commons.wikimedia.org/wiki/File:ASCII_Code_Chart.svg#/media/Fayl:ASCII_Code_Chart.svg (viitattu 24. 06. 2020).

[22] *Dynamic Host Configuration Protocol*. Bucknell University. 1997. URL: https://tools.ietf.org/html/rfc2131.

[23] *DHCP Options and BOOTP Vendor Extensions*. Bucknell University. 1997. URL: https://tools.ietf.org/html/rfc2132.

[24] *Understanding and Troubleshooting DHCP in Catalyst Switch or Enterprise Networks*. Cisco Inc. 2008. URL: https://www.cisco.com/c/en/us/support/docs/ip/dynamic-address-allocation-resolution/27470-100.pdf.

[25] *Automate Device Provisioning with Cisco IOS XE Zero Touch Provisioning*. Cisco Inc. 2019. URL: https://blogs.cisco.com/developer/device-provisioning-with-ios-xe-zero-touch-provisioning.

[26] *Cisco DNA Center User Guide, Release 1.2.8*. 2. joulukuuta 2019. URL: https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/dna-center/1-2-8/user_guide/b_dnac_ug_1_2_8/b_dnac_ug_1_2_8_chapter_01100.html (viitattu 24. 06. 2020).

[27] *Solution Guide for Cisco Network Plug and Play*. 17. toukokuuta 2018. URL: https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Plug-

and - Play / solution / guidexml / b _ pnp - solution - guide . html (viitattu 24. 06. 2020).

[28] *S5700 V100R006C00 Configuration Guide - Device Management*. 16. lokakuuta 2012. URL: https://support.huawei.com/enterprise/en/doc/EDOC0100533703?section=j004 (viitattu 24. 06. 2020).

[29] *S600-E V200R011C10 Configuration Guide - Basic Configuration*. 23. huhtikuuta 2020. URL: https://support.huawei.com/enterprise/en/doc/EDOC1000178013/48bdb5bb/ (viitattu 24. 06. 2020).

[30] *Raspberry Pi GPIO Programming in C*. 26. toukokuuta 2018. URL: https://www.bigmessowires.com/2018/05/26/raspberry-pi-gpio-programming-in-c/ (viitattu 24. 06. 2020).

[31] *RPi Low-level peripherals*. 28. kesäkuuta 2019. URL: https://elinux.org/RPi_Low-level_peripherals#Power_pins (viitattu 24. 06. 2020).

[32] *MAX32323-V to 5.5-V MultichannelRS-232Line Driver/ReceiverWith 15-kV ESD Protection*. 1. kesäkuuta 2017. URL: http://www.ti.com/lit/ds/symlink/max3232.pdf (viitattu 24. 06. 2020).

[33] *Raspberry Pi boot modes*. 24. kesäkuuta 2020. URL: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/README.md (viitattu 24. 06. 2020).

[34] *The Kernel Command Line*. 24. kesäkuuta 2020. URL: https://www.raspberrypi.org/documentation/configuration/cmdline-txt.md (viitattu 24. 06. 2020).

[35] *UART configuration*. 24. kesäkuuta 2020. URL: https://www.raspberrypi.org/documentation/configuration/uart.md (viitattu 24. 06. 2020).

[36] *Wiring Pi - The GPIO utility*. 24. kesäkuuta 2020. URL: http://wiringpi.com/the-gpio-utility/ (viitattu 24. 06. 2020).

[37] *PySerial 3.0 documentation*. 24. kesäkuuta 2020. URL: https://pyserial.readthedocs.io/en/latest/ (viitattu 24. 06. 2020).